Środowisko programisty





UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt "Programowa i strukturalna reforma systemu kształcenia na Wydziale Mat-Fiz-Inf". Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Człowiek-najlepsza inwestycja

Uniwersytet Marii Curie-Skłodowskiej Wydział Matematyki, Fizyki i Informatyki Instytut Informatyki

Środowisko programisty

Grzegorz M. Wójcik Sławomir Kotyra



LUBLIN 2011

© Copyright by Instytut Informatyki UMCS Lublin 2011

Grzegorz M. Wójcik Sławomir Kotyra ŚRODOWISKO PROGRAMISTY

Recenzent: Andrzej Bobyk

Opracowanie techniczne: Marcin Denkowski Projekt okładki: Agnieszka Kuśmierska

Praca współfinansowana ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Publikacja bezpłatna dostępna on-line na stronach Instytutu Infromatyki UMCS: informatyka.umcs.lublin.pl.

Wydawca

Instytut Informatyki Uniwersytet Marii Curie-Skłodowskiej w Lublinie pl. Marii Curie-Skłodowskiej 1, 20-031 Lublin Redaktor serii: prof. dr hab. Paweł Mikołajczak www: informatyka.umcs.lublin.pl email: dyrii@hektor.umcs.lublin.pl

Druk

ESUS Agencja Reklamowo-Wydawnicza Tomasz Przybylak ul. Ratajczaka 26/8 61-815 Poznań www: www.esus.pl

ISBN: 978-83-62773-18-3

SPIS TREŚCI

| W | ⁷ STĘP | ix |
|---|--|--|
| D | EFINICJE | xi |
| 1 | ŻYCIE W ŚWIECIE GNU/LINUX1.1. Wprowadzenie1.2. Historia GNU/Linux w pigułce1.3. Dlaczego Ubuntu?1.4. Trzech wspaniałych czyli ludzie GNU/Linux - biogramy1.5. Zamiast podsumowania1.6. Zadania | 1 2 3 5 8 11 12 |
| 2 | INSTALACJA I KONFIGURACJA UBUNTU 10.04 LTS2.1. Wprowadzenie2.2. Przygotowanie do instalacji2.3. Proces instalacji2.4. Strojenie systemu2.5. Podsumowanie2.6. Zadania | 15 16 16 17 24 28 29 |
| 3 | INTERFEJS WIERSZA POLECEŃ 3.1. Podstawy interfejsu wiersza poleceń | 31 32 38 41 |
| - | 4.1. Operacje na plikach 4.2. Dowiązania do plików 4.3. Atrybuty plików 4.4. Programy do zarządzania plikami | 42 52 54 57 |
| 5 | PRACA Z PLIKIEM TEKSTOWYM 5.1. Pliki tekstowe 5.2. Zaawansowana obsługa plików tekstowych | 69 70 73 |

| | 5.3. Podstawowe edytory plików tekstowych | \$6 |
|----|--|-----------|
| 6 | Skrypty powłoki bash 10 |)5 |
| | 6.1. Wstep $\dots \dots \dots$ |)6 |
| | 6.2. Uruchamianie skryptów powłoki |)6 |
| | 6.3. Komentarze \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots 10 |)7 |
| | 6.4. Zmienne i argumenty wywołania skryptu |)7 |
| | 6.5. Operacje arvtmetyczne |)8 |
| | 6.6. Operacje logiczne |)8 |
| | 6.7. Sterowanie przebiegiem wykonania skryptu | 0 |
| | 6.8. Pliki inicjalizacyjne powłoki bash | 5 |
| | $6.9. \text{ Podsumowanie} \qquad \dots \qquad $ | 7 |
| 7 | | 10 |
| 1 | F OLĄCZENIA ZDALNE 11 7 1. Desten de edlembere komputere 12 | 19 |
| | 7.1. Dostęp do odlegiego komputera | 20 20 |
| | 7.2. Przesyłanie pikow mędzy komputerami | |
| | 7.3. Połączenia z systemu windows | :D |
| 8 | Edytor EMACS 13 | 31 |
| | 8.1. Wprowadzenie | 52 |
| | 8.2. Instalacja i konstruowanie poleceń edytora | 33 |
| | 8.3. Podstawy edycji | 6 |
| | 8.4. Dostosowywanie edytora | 37 |
| | 8.5. Podsumowanie | 39 |
| | 8.6. Zadania | 9 |
| 9 | Wykresy funkcji i wiznalizacja danych | |
| 0 | w środowisku Gnuplot 14 | 11 |
| | 9.1 Wprowadzenie 14 | 2 |
| | 9.2 Wykresy funkcji 14 | 3 |
| | 9.3 Wizualizacia danych 14 | 17 |
| | 9.4 Tworzenie skryptów dla Gnuplota | 51 51 |
| | 9.5 Podsumowanie | 52 |
| | 9.6 Zadania 15 | 52 |
| | | |
| 10 | System składu IAT $_{ m E}$ X 15 | 55 |
| | 10.1. Wprowadzenie | 6 |
| | 10.2. Instalacja systemu LATEX w Ubuntu | 67 |
| | 10.3. Struktura prostego dokumentu | 68 |
| | 10.4. Wyrażenia matematyczne | 51 |
| | 10.5. Podsumowanie | 52 |
| | 10.6. Zadanja | 53 |

| 11 System tworzenia prezentacji Beamer | 167 |
|--|-----|
| 11.1. Wprowadzenie | 168 |
| 11.2. Szkielet prezentacji | 168 |
| 11.3. Formatowanie tekstu | 171 |
| 11.4. Bloki, listy, tabele i rysunki | 171 |
| 11.5. Podsumowanie | 176 |
| 11.6. Zadania | 176 |
| Zakończenie | 179 |
| Spis rysunków | 180 |
| Bibliografia | 183 |



Wstęp

Niniejszy skrypt stanowi propozycję dla studentów pierwszych lat studiów informatycznych i kierunków pokrewnych. Przedstawiono w nim środowisko programisty z punktu widzenia użytkownika systemu operacyjnego Linux. Treści zawarte w skrypcie wyczerpują standardy wymagań określone sylabusami dla wykładów wprowadzających początkujących użytkowników w świat programowania i administracji systemami uniksowymi.

Tuż po Wstępie zamieszczono podstawowe definicje i terminologię, którą autorzy posługują się w dalszych częściach skryptu.

W rozdziale pierwszym przedstawiono historię systemów z rodziny Unix i Linux, zamieszczono przegląd najważniejszych dystrybucji.

Rozdział drugi przybliża użytkownikowi proces instalacji systemu operacyjnego Ubuntu 10.04 w wersji desktop.

W trzecim rozdziale czytelnik zostaje zapoznany z podstawami obsługi wiersza poleceń systemu Linux. Ponadto otrzymuje informację dotyczące kontroli przepływu strumieni w systemie.

Rozdział czwarty traktuje o zaawansowanej pracy z plikami w systemie. Przedstawiamy podstawy zarządzania plikami z poziomu wiersza poleceń jak również zaawansowany program do obsługi plików mc. Z tego rozdziału czytelnicy dowiadują się o mechanizmach tworzenia dowiązań do plików oraz o ich atrybutach.

W piątym rozdziale szczegółowo omówiono pracę z plikami tekstowymi. Zaproponowano edycję plików z wykorzystaniem prostych edytorów: vi, nano, oraz mcedit.

Rozdział szósty można traktować jako kurs programowania powłoki systemowej bash. Oprócz operacji arytmetycznych i logicznych wykonywanych na poziomie powłoki przedstawiono składnię instrukcji warunkowej, wielokrotnego wyboru oraz trzech rodzajów pętli.

W rozdziale siódmym opisano możliwości zdalnego połączenia z maszynami uniksowymi zarówno z poziomu systemu Linux jak i Windows.

Rozdział ósmy to opis podstawowych funkcji edytora Emacs.

Przedstawiamy fundamentalne polecenia edytora, skróty klawiszowe oraz opisujemy dwa przykłady przystosowania Emacsa do własnych potrzeb.

W rozdziale dziewiątym zapoznajemy czytelników z możliwościami wizualizacji danych w środowisku Gnuplot. Przedstawiamy wizualizacje funkcji jednej zmiennej, dwóch zmiennych oraz zbiorów danych doświadczalnych na wykresach kilku typów.

Rozdział dziesiąty przedstawia zalety korzystania z systemu składu tekstu LATEX. Zapoznajemy czytelników z procesem instalacji środowiska TEXw systemie. Następnie pokazujemy jak stworzyć prosty dokument o wielopoziomowej strukturze.

W rozdziale jedenastym opisano podstawowe właściwości klasy Beamer wykorzystywanej przez IAT_FXdo tworzenia zaawansowanych prezentacji.

Ponadto niektóre rozdziały zostały opatrzone zestawami przykładowych zadań, które mogą być zrealizowane w ramach ćwiczeń w laboratoriach.

Definicje

W celu ujednoznacznienia terminologii, jaka będzie używana w niniejszym skrypcie, jest konieczne zdefiniowanie pojęć podstawowych. Definicje będą miały prosty, niemal intuicyjny charakter, ponieważ celem jest jedynie uniknięcie nieporozumień.

Definicje terminów podstawowych

Przymiotnik *programowalny* określa cechę urządzenia, którego sposób działania nie jest jednoznacznie określony. Zachowanie urządzenia jest zależne od sposobu zaprogramowania. Dwa identyczne urządzenia różnie zaprogramowane mogą wykazywać (i najczęściej wykazują) odmienne zachowanie.

Polecenie lub *instrukcja* to pojedyncza, nie dająca się podzielić na prostsze operacja (czynność), wykonywana przez urządzenie programowalne.

Program to zestaw poleceń lub instrukcji, jakie może wykonać urządzenie programowalne. Program determinuje (jednoznacznie określa) sposób działania urządzenia programowalnego.

Pod pojęciem *systemu komputerowego* lub w skrócie *komputera* należy rozumieć kompletne środowisko sprzętowe, zdolne do zaprogramowania (czyli programowalne) i wykonywania programu. Komputer posiada ponadto szereg innych właściwości, jednak w tym miejscu są dla nas ważne dwie wymienione.

Elementem komputera faktycznie wykonującym program jest procesor. *Procesor* to urządzenie wykonane w sposób umożliwiający pobranie pojedynczej instrukcji do wykonania oraz jej wykonanie. Te dwie realizowane cyklicznie czynności (pobierz następną instrukcję, wykonaj) są wystarczające, by procesor spełniał swoje zadanie.

Zestaw wszystkich instrukcji możliwych do wykonania przez procesor (lub bardziej intuicyjnie "rozumianych przez procesor") jest nazywany *językiem maszynowym*. Procesor może wykonywać wyłącznie programy w języku (kodzie) maszynowym. Język programowania – język wzorowany na języku naturalnym, jednak spełniający rygorystyczne wymagania formalne, tak by było możliwe przetworzenie go na język maszynowy.

Kod źródłowy – tekst programu napisanego w jednym z języków programowania. Kod źródłowy bywa też nazywany programem źródłowym lub w skrócie programem. Ostatnie określenie implikuje niejednoznaczność z wcześniejszą definicją terminu program, jednak kontekst wypowiedzi z reguły nie pozostawia wątpliwości, czy należy mieć na myśli kod źródłowy, czy program w postaci wykonywalnej.

Narzędzie programistyczne – program wspomagający szeroko rozumianą realizację zadań związanych z tworzeniem innych programów albo produktów pokrewnych.

System operacyjny – zestaw programów tworzących jednolite środowisko uruchamiania innych programów oraz udostępniający funkcje świadczone przez system operacyjny. Na systemie operacyjnym spoczywa odpowiedzialność za znajomość szczegółów realizacji sprzętowej systemu komputerowego. W ten sposób wersje wykonywalne programów mogą być przygotowywane w pewnym oderwaniu od znajomości sprzętu i łatwo przenoszone w obrębie komputerów pracujących pod kontrolą zgodnych ze sobą systemów operacyjnych. Przenoszenie programów pomiędzy różnymi systemami operacyjnymi wymaga przygotowania nowej wersji wykonywalnej programu dla innego systemu operacyjnego.

W otwartych systemów przypadku operacyjnych, zestaw oprogramowania tworzący rozpowszechnianą całość nazywany jest dystrybucją. Dystrybucja może występować w wielu wersjach. W przypadku systemów zamkniętych funkcjonuje tylko termin wersja.

Program uruchomiony i działający pod kontrolą systemu operacyjnego jest nazywany *procesem*. System operacyjny jest odpowiedzialny za zarządzanie procesami.

Maszyna wirtualna – program tworzący warstwę pośrednią pomiędzy programem i systemem operacyjnym albo (rzadko) programem i komputerem. Maszyna wirtualna tworzy środowisko uruchomieniowe dla programów w postaci kodu bajtowego, przygotowanych dla tejże maszyny wirtualnej. W przypadku realizacji maszyny wirtualnej dla różnych systemów operacyjnych lub różnych platform sprzętowych, możliwe jest bezpośrednie (nie wymagające dodatkowego przygotowania) przenoszenie kodu bajtowego pomiędzy nimi.

Interpreter – program wykonujący kod źródłowy albo kod bajtowy. Program (w postaci kodu źródłowego albo bajtowego) jest przetwarzany na instrukcje procesora podczas każdego wykonania.

Kompilator – narzędzie programistyczne służące do przetwarzania kodu źródłowego, napisanego w jednym z języków programowania, na kod

maszynowy albo kod bajtowy. Proces przetwarzania kodu źródłowego nazywa się *kompilacją*. Każda zmiana w kodzie źródłowym wymaga ponownej kompilacji.

Plik – ciąg danych przechowywany w urządzeniach pamięci masowej (dyskach magnetycznych, optycznych, pamięciach typu flash itp.). Patrząc od strony reprezentacji fizycznej dane te są bajtami, czyli ośmiobitowymi liczbami binarnymi (dwójkowymi) z przedziału 0..255. Nie wdając się w szczegóły techniczne pamięć masowa jest urządzeniem blokowym, a system plików systemu operacyjnego zapewnia logiczny porządek przechowywanych na niej danych.

Jeden rodzaj plików ma szczególne znaczenie. Są to tak zwane *katalogi* (w systemie Windows nazywane *folderami*), czyli pliki których zawartość system operacyjny potrafi zapisać, a później odczytać i zinterpretować. W katalogach przechowywane są między innymi nazwy i atrybuty plików (w tym również katalogów podrzędnych), czas dostępu i czas ostatniej modyfikacji, rozmiar pliku i jego fizyczne położenie na dysku, informacje o właścicielu.

Katalog bieżący albo *katalog roboczy* – katalog w systemie plików, do którego odnoszą się wydawane polecenia. Katalog bieżący może być łatwo i wielokrotnie zmieniany odpowiednimi poleceniami.

Zmienna środowiskowa – napis przechowywany w obszarze pamięci interpretera poleceń. System operacyjny inicjuje wiele zmiennych środowiskowych podczas uruchamiania interpretera poleceń. Użytkownik ma możliwość korzystania z istniejących zmiennych (włączając w to modyfikowanie), a także deklarowania własnych zmiennych.

Przez środowisko programistyczne (środowisko programisty) należy rozumieć zestaw narzędzi programowych umożliwiających tworzenie, uruchamianie, testowanie programu, a także narzędzi pomocniczych umożliwiających zarządzanie projektem programistycznym np. w sensie kontroli wersji lub tworzenia dokumentacji.

W szczególnych przypadkach, nie rozważanych w tym opracowaniu, termin środowisko programistyczne może również obejmować specyfikację platformy sprzętowej, na jakiej zachodzi proces tworzenia oprogramowania lub innego produktu pokrewnego (na przykład strony internetowej w języku HTML).

Unix– system operacyjny opracowany w 1969 r. w Bell Labs przez Dennisa Ritchie i Kena Thompsona.

Linux– jądro wielu dystrybucji otwartych uniksopodobnych systemów operacyjnych.

GNU– zestaw programów tworzących łącznie z jądrem (np. Linux)kompletny system operacyjny (GNU/Linux),o cechach użytkowych wzorowanych na Uniksie.

Ponieważ system operacyjny GNU/Linux jest rozpowszechniany z kodem źródłowym, narzędziami programistycznymi i dokumentacją, można zaryzykować stwierdzenie, że jest on kompletnym środowiskiem programistycznym nastawionym na rozwój samego siebie.

Rozdział 1

Życie w świecie GNU/Linux

| 1.1. | Wprowadzenie | 2 |
|------|--|----------|
| 1.2. | Historia GNU/Linux w pigułce | 3 |
| 1.3. | Dlaczego Ubuntu? | 5 |
| 1.4. | Trzech wspaniałych czyli ludzie GNU/Linux - biogramy | 8 |
| 1.5. | Zamiast podsumowania | 11 |
| 1.6. | Zadania | 12 |
| | | |

1.1. Wprowadzenie

Systemy operacyjne typu Unix/Linux od kilku dekad kojarzą się z solidnością, stabilnością i bezpieczeństwem. Mimo to nie zyskały wystarczającej popularności wśród przeciętnych użytkowników komputerów osobistych. Powodów mogło być wiele: przede wszystkim z założenia systemy UNIX nie były projektowane do zapewnienia rozrywki lub obsługi multimedialnej rozmaitych urządzeń, z kolei systemy z rodziny Linux przynajmniej na początku mogły dostarczać niedoświadczonym użytkownikom niemałej porcji trudności, a przemysł gier komputerowych i innego oprogramowania użytkowego w wyniku pewnych ekonomicznych zbiegów okoliczności zaczął koncentrować się wokół systemu operacyjnego Microsoft Windows.

Z drugiej strony wszędzie tam gdzie uprawiano nauki techniczne i ścisłe - systemy klasy Unix wiodły i wciąż wiodą prym. Przytłaczająca większość superkomputerów z listy TOP 500 pracuje pod kontrolą Unix/Linux (89% Linux, 10% Unix, mniej niż 1% Windows [1]). W Hollywood firmy takie jak Disney/Pixar, DreamWorks Animation, Sony, ILM, by poprzestać na wyliczeniu najważniejszych, używają Linuksa do prawie wszystkich superprodukcji. Szacuje się, że Linux jest zainstalowany na 95% tamtejszych komputerów osobistych i serwerów [2]. Trudno też wyobrazić sobie funkcjonowanie współczesnego Internetu bez systemu Linux, który obsługuje zdecydowaną większość serwerów sieciowych. W połowie 2010. roku firma Google zabroniła swoim pracownikom korzystania z komputerów z zainstalowanym systemem Windows z powodu niestabilności i luk bezpieczeństwa. Jako alternatywę pracownicy otrzymali systemu Linux i Mac OS X [3].

W ostatnich latach można zaobserwować bardzo pozytywne zjawisko uczłowieczania Linuksa w taki sposób, by mógł on stanowić w pełni funkcjonalny system operacyjny do zastosowań domowych, rozrywkowych i multimedialnych. W chwili obecnej nie istnieje obszar informatyczny, w którym nie dałoby się sprawnie poruszać wykorzystując Linuksa. Niestety bardzo często wybór systemu operacyjnego jest kwestią przyzwyczajenia albo wręcz narzucenia z góry przez producentów sprzętu komputerowego.

Tym nie mniej studenci informatyki z pewnością nie są ani przeciętnymi ani niedoświadczonymi użytkownikami komputerów osobistych. Wielu z nich już za kilka lat stanowić będzie rdzeń elity europejskiego społeczeństwa informacyjnego w realiach XXI wieku. Dlatego znajomość systemu Linux nie powinna być dla nich kwestią mody albo wypadkową stwierdzenia, że "tak wypada", lecz absolutną koniecznością.

Należy jednak podkreślić, iż obcowanie z Linuksem stanowi niemałą przyjemność. Linuksowa subkultura wytworzyła specyficzną społeczność,

coś na kształt filozofii, ideologię która pozwala w świecie dominującej technologii dokonywać właściwych informatycznych wyborów. Coraz częściej można usłyszeć o kulturze hakerskiej, społeczność organizuje zloty, wspólne przedsięwzięcia, happeningi. Nie pozostaje zatem nic innego jak zachęcić czytelników do radosnego rozpoczęcia przygody w świecie wolnego i otwartego oprogramowania.

1.2. Historia GNU/Linux w pigułce

W czasach zamierzchłych dla większości czytelników, to jest w latach sześćdziesiątych dwudziestego wieku w MIT, AT&T Bell Labs oraz w laboratoriach firmy general Electric rozwijano system operacyjny Multics przeznaczony dla komputera Mainframe GE-645. Multics wprowadzał jak na swoje czasy wiele innowacji, ale niestety również sporo problemów. Właściwości zdefiniowane dla nowego systemu były dobre, jednak sam system wykazywał zbyt duży poziom złożoności, co doprowadziło Bell Labs do wycofania z projektu.

Zespół, który do niedawna zajmowali się tworzeniem Multicsa zaangażował się teraz niemal hobbystycznie, bez finansowego wsparcia Bell Labs w stworzenie mniej skomplikowanego, hierarchicznego systemu operacyjnego o nazwie Unics, która z biegiem lat wyewoluowała do dziś używanego Unix. Za ojców Uniksa uważa się Kena Thompsona i Dennisa Ritchiego (Rys. 1.1), którzy w roku 1969 uruchomili pierwszą jego wersję. System charakteryzował się zdolnością kontrolowania procesów, plikami stowarzyszonymi z urządzeniami i interpretatorem linii poleceń. Dopiero na początku lat siedemdziesiątych Thompson i Ritchie zobowiązali się napisać Uniksa na szybsze komputery typu PDP-11/20 co zapewniło im wsparcie finansowe AT&T i umożliwiło dalszy rozwój. Lata siedemdziesiąte to czas wdrażania komercyjnego Uniksa w amerykańskich uniwersytetach i laboratoriach badawczych.

W 1983 roku Richard Stallman i założona przez niego Fundacja Wolnego Oprogramowania (FSF - Free Software Foundation) opublikowali licencję GNU (Inspirowana piosenką "The Gnu" grupy Flanders and Swann nazwa tworzy rekurencyjny skrót "GNU is not UNIX". Jako logo idei wybrano antropomorficzny wizerunek antylopy Gnu.). Licencja głosiła ideę wolnego oprogramowania (to znaczy każdy kto otrzymuje oprogramowanie ma całkowitą dowolność w używaniu, studiowaniu, modyfikowaniu i dystrybucji tak licencjonowanego oprogramowania). Zamierzeniem FSF było stworzenie wolnego systemu operacyjnego w rozumieniu GNU przypominającego działaniem system Unix.



Rysunek 1.1: Ken Thompson (z lewej) i Dennis Ritchie (z prawej)

W latach osiemdziesiątych system GNU posiadał już pokaźny zestaw bibliotek zgodny z tymi używanymi w systemie Unix, doskonały edytor tekstu Emacs (Editor MACroS) oraz bardzo dobry kompilator gcc (GNU Compiler Collection). Na początku lat dziewięćdziesiątych stworzono środowisko graficzne X-Window System oraz system składu drukarskiego $T_{\rm E}X$. Projekt jądra systemu GNU Hurd po dziś dzień nie został jeszcze sfinalizowany (chociaż istnieje eksperymentalna wersja Debiana oparta o to jądro).

Jednak w 1991 roku fiński student Linus Torvalds opublikował jądro systemu operacyjnego Linux oparte o licencję GNU General Public License. Doprowadziło to do stworzenia systemu operacyjnego GNU/Linux, który oprócz jądra napisanego przez Torvaldsa zawierał cały zestaw pakietów GNU. Obecnie Linux zajmuje większą część informatycznej przestrzeni zarezerwowanej wcześniej dla Uniksa. Filozofia licencjonowania pozwoliła na stworzenie setek dystrybucji, mniej lub bardziej popularnych, jednak zawsze przestrzegających filozofii wolności oprogramowania i otwartości kodu źródłowego. Najważniejsze jednak jest to, że zupełnie za darmo można cieszyć się w pełni funkcjonalnym, multimedialnym systemem operacyjnym.

Systemy uniksowe trwale i na długo wpisały się w przeszłość teraźniejszość i przyszłość informatyki. 1 stycznia 1970 roku uznawany jest za początek ery lub epoki uniksowej. Związana jest z tym pewna ciekawostka, ale i zagrożenie. Otóż system Unix i jego pochodne przechowują czas w zmiennej systemowej wyrażającej się jako liczba

sekund od 1 stycznia 1970 roku (tak zwana Unix Epoch). Zmienna $time_t$ typu signed integer 32 przekroczy swój zakres 19 stycznia 2038 roku. Może to spowodować prawdziwą katastrofę, o niepomiernie większych skutkach niż nagłośniona przez dziennikarzy katastrofa roku 2000.

1.3. Dlaczego Ubuntu?

Jądro systemu Linux wraz z zestawem pakietów bibliotek i oprogramowania, które po zainstalowaniu stanowią działający system operacyjny przyjęto nazywać dystrybucją. Istnieje wiele dystrybucji systemu Linux, z reguły wydawane są one przez firmy rozwijające oprogramowanie, zespoły pasjonatów, instytucje państwowe i uniwersytety. Przy odrobinie zaangażowania można pokusić się o stworzenie własnej dystrybucji.

Przedstawimy opis kilku najpopularniejszych dystrybucji Linuksa oraz motywację, która przemawia za wyborem dystrybucji Ubuntu.

Slackware - najstarsza z wciąż utrzymywanych dystrybucji Linuksa. Projekt kierowany przez Patricka Volkerdinga zaowocował wydaniem pierwszej wersji systemu 17 lipca 1993 roku. Regułą, która wydaje się być nadrzędną w budowaniu dystrybucji Slackware jest sformułowana przez twórców Uniksa idea KISS (Keep It Simple, Stupid! - Nie komplikuj, głupcze!). Przejrzysta dystrybucja oferuje najnowsze oprogramowanie, utrzymuje dobrą linuksową tradycję, a większość aspektów konfiguracyjnych polega na edycji odpowiednich plików tekstowych. Ta prostota i przejrzystość może okazać się wadą dystrybucji dla wszystkich tych, którzy rozpoczynają przygodę z Linuksem [4, 5].

Debian - to również jedna z najstarszych, ale i najlepszych dystrybucji Linuksa, datowana na 16 sierpnia 1993 r. Projekt Debian to związek pasjonatów indywidualistów pragnących zbudować wolny, uniksowy system operacyjny. Największą zaletą Debiana jest wyjątkowa stabilność oferowanych przezeń ponad dwudziestu tysięcy pakietów. Z utrzymaniem stabilności niestety związana są rzadko pojawiające się nowe wersje systemu. Zatem Debian jest niezastąpiony wszędzie tam gdzie zależy nam na stabilności i bezpieczeństwie, a nie na najnowszych wersjach oprogramowania [4, 6].

Fedora - dawniej Red Hat to jedna z najpopularniejszych dystrybucji Linuksa. Fedora wciąż jest ściśle związana z Red Hatem, który po komercjalizacji zajmuje się wydawaniem szczególnie stabilnych, ale odpłatnych dystrybucji na potrzeby szeroko rozumianego biznesu. Zaletą Fedory jest elastyczność działania i łatwość konfiguracji systemu z wykorzystaniem programów pracujących w trybie graficznym. Inżynierowie Red Hata zapewniają wsparcie drużynie tworzącej Fedorę, [4, 7].

CentOS - to bardzo dobry system do zastosowań sieciowych. Ponieważ komercyjny Red Hat Enterprise Linux wciąż pozostawia otwarte kody źródłowe swojego oprogramowania na zasadach licencji GNU/GPL twórcy CentOSa kompilują te kody tworząc darmową, w założeniu w 100% zgodną z Red Hatem dystrybucję systemu operacyjnego [4, 8]. Nic dodać nic ująć.

Mandriva - dawniej francuski Mandrake (połączony z brazylijskim Linuksem Conectiva) to dystrybucja o filozofii podobnej do Fedory, oparta o prostotę działania i wygodny interfejs użytkownika, łatwe możliwości konfiguracji w większości w trybie graficznym. Celem twórców dystrybucji było stworzenie poczucia łatwości korzystania z Linuksa [4, 9].

openSUSE - niemiecki Linux, a właściwie projekt społecznościowy sponsorowany przez firmę Novell. Podobnie jak Fedora współpracuje z Red Hatem, tak openSUSE jest jak się uważa siostrzaną dystrybucją komercyjnej SUSE Linux Enterprise przeznaczonej dla biznesowych zastosowań. Wsparcie firmy Novell stanowi istotną zaletę systemu, ponadto od lat SUSE uchodzi za jeden z najbardziej graficznie dopracowanych systemów. Niestety ciągnie to za sobą dość spore jak na Linuksa wymagania sprzętowe, co może zniechęcać wielu potencjalnych użytkowników [4, 10].

Gentoo - to wyższa szkoła jazdy i jedna z najbardziej fascynujących dystrybucji. Idea Gentoo polega na tym, że system operacyjny jest kompilowany ze źródeł dla konkretnej maszyny danego użytkownika. Pomijając długi proces kompilacji, instalacja Gentoo wymaga niemałej wiedzy i doświadczenia. Radzimy spróbować zainstalować Gentoo na wolnym komputerze. Nawet jeśli się nie uda, wiedza jaką zdobędą śmiałkowie jest nie do przecenienia. Posiadając dobrze zainstalowane Gentoo mamy pewność, że system działa możliwie najszybciej na konkretnym sprzęcie [4, 11].

Arch - to dystrybucja rozwijana dla doświadczonych i kompetentnych użytkowników Linuksa. Cechą charakterystyczną jest prawdopodobnie najszybszy dostęp do najnowszych wersji pakietów zorganizowany dzięki systemowi zarządzana *pacman*. Przygodę z Arch rozpoczynamy od instalacji bazowej, a całość dystrybucji musimy zorganizować sobie

sami. Dlatego Arch może okazać się niezbyt dobrym wyborem dla początkujących użytkowników [4, 12].

PCLinuxOS - to zdobywająca sobie coraz większą popularność dystrybucja Linuksa, w której postawiono na multimedia. Ten Linux obsłuży prawie na pewno każdą nawet najnowszą kartę grafiki albo dźwiękową. W tym Linuksie najłatwiej obejrzeć film, posłuchać muzyki, zgrać zdjęcia z aparatu. PCLinuxOS jest dobry dla początkujących, którym zależy na bezproblemowej przesiadce z systemu Windows do uniksowego świata [4, 13].

Ubuntu - Ubuntu jest afrykańską ideologią oznaczającą w skrócie "humanizm dla innych"[14]. Wyraża "wiarę w uniwersalne więzy oparte na chęci dzielenia się z innymi, które łączą całą ludzkość". To wywodząca się z Debiana, ale w pełni niezależna najpopularniejsza obecnie dystrybucja Linuksa. Wsparcie dla Ubuntu oferują zarówno społeczność jak i profesjonaliści z firmy Canonical. Społeczność Ubuntu buduje się wokół Manifestu Ubuntu, który zakłada, że:

- Każdy użytkownik komputera powinien móc w sposób całkowicie wolny uruchamiać, rozpowszechniać, analizować, zmieniać i ulepszać oprogramowanie, którym się posługuje bez konieczności ponoszenia jakichkolwiek opłat licencyjnych,
- Każdy użytkownik komputera powinien mieć możliwość używania oprogramowania we właściwej mu wersji językowej,
- Każdy użytkownik komputera powinien móc używać dowolnego oprogramowania bez względu na swoją ewentualną niepełnosprawność oraz bez względu na charakter tej niepełnosprawności.

Definicja wolnego oprogramowania wywodzi się z ideologii GNU i w ujęciu Ubuntu wolne oprogramowanie to takie, które oferuje cztery wolności:

- --wolność używania oprogramowania bez względu na cel,
- wolność analizowania sposobu działania oprogramowania i dostosowywania go do swoich potrzeb,
- wolność rozpowszechniania kopii oprogramowania by pomóc innym,
- wolność udoskonalania oprogramowania oraz rozpowszechniania własnych poprawek aby również inni mogli z nich korzystać.

Ubuntu pozostanie zawsze wolne i zawsze za darmo, firma Canonical dwa razy w roku wydaje kolejne wersje systemu operacyjnego zarówno na biurko jak i z przeznaczeniem na serwer. Otrzymujemy środowisko równie stabilne jak Debian, przy czym pakiety zazwyczaj są w prawie najnowszej wersji. Pozostaje nam łatwość i przejrzystość konfiguracji Debiana w trybie tekstowym, ale mniej doświadczeni użytkownicy mogą równie efektywnie korzystać z narzędzi graficznych. "Humanitaryzm ukierunkowany na innych" albo "Linux dla ludzi" to hasła, za którymi stoi sens Ubuntu. Ta dystrybucja powinna zostać wybrana przez wszystkich, którzy po prostu chcą wykonać swoją pracę bez zbędnych komplikacji i niepotrzebnego zamieszania. Nie ma już zatem znaczenia, że zaawansowany technologicznie Linux jest skomplikowany. Z dniem wypuszczenia pierwszej wersji Ubuntu w świat dokonała się swoista zmiana, bo w tej dystrybucji przeciętny użytkownik mógł poruszać się z łatwością równą interfejsowi Windows. Dysponował jednak o wiele bardziej stabilnym system operacyjnym z niepomiernie szerszym wachlarzem potencjalnych zastosowań [4, 14].

Cechy Ubuntu, które powinny przekonać użytkowników do wyboru dystrybucji:

- Regularne i aktualne wydania.
- Ukierunkowanie na jakość.
- Wsparcie społeczności i komercyjne.
- Łatwe uzyskiwanie programów i uaktualnień.
- Ukierunkowanie na użyteczność.
- Ukierunkowanie na internacjonalizację.
- Aktywna i zaangażowana społeczność.

Większość dystrybucji boryka się z podobnymi problemami, ale tylko Ubuntu posiada jedyny w swoim rodzaju sposób ich rozwiązywania. Każdy kto decyduje się na Ubuntu zyskuje pewność, że gdy wpadnie w tarapaty z systemem operacyjnym w pobliżu znajdzie się ktoś, kto poda rękę. Firma Canonical, która zdążyła zaskarbić sobie zaufanie milionów użytkowników jest również gwarantem ciągłości dystrybucji przynajmniej przez najbliższe lata [14].

1.4. Trzech wspaniałych czyli ludzie GNU/Linux - biogramy

Richard Stallman

Richard Matthew Stallman (Rys. 1.2), znany jako rms, urodzony 16 marca 1953 roku w Nowym Jorku na Manhattanie, haker i jeden z twórców ruchu wolnego oprogramowania [15]. W 1974 roku ukończył fizykę na Uniwersytecie Harvarda, pracował jako haker w Laboratorium Sztucznej Inteligencji MIT.

W latach osiemdziesiątych pod wpływem prężnie rozwijającego się systemu korporacji programistycznych Stallman porzucił pracę na MIT stając w obronie wolnego oprogramowania i kultury hakerskiej. W 1983 roku opublikował Manifest GNU, wkrótce potem założył Free Software Foundation. Jest współautorem wielu programów wchodzących w skład



Rysunek 1.2: Richard Matthew Stallman (rms)

pakietu GNU, w tym kultowego edytora Emacs oraz kompilatora gcc. Postać kontrowersyjna, znana z ciętego języka, specyficznych poglądów politycznych i moralnych.

Linus Torvalds

Linus Benedict Torvalds (Rys. 1.3), urodzony 28 grudnia 1969 roku w Helsinkach, fiński programista, twórca jądra systemu Linux [16]. Studiował na Uniwersytecie Helsińskim, tworząc jądro przyczynił się do gwałtownego rozkwitu ruchu Wolnego Oprogramowania. Mieszka w Santa Clara w Kalifornii, pracuje w Linux Foundation, ma żonę i dwie córki.

Zainspirowany edukacyjnym systemem Minix Linus poczuł potrzebę stworzenia wydajnego systemu operacyjnego typu Unix i takiego, który mógłby zostać uruchomiony na jego domowym komputerze PC. Pierwsza wersja jądra została opublikowana 15 października 1991. Obecnie Torvalds zajmuje się tylko nadzorowaniem kolejnych pojawiających się wersji. Maskotką Torvaldsa, a przez to i całego szeroko rozumianego Linuksa wszystkich dystrybucji jest pingwin o imieniu Tux (Rys. 1.4).

Mark Shuttleworth

Mark Shuttleworth (Rys. 1.5), urodzony 18 września 1973 w Welkom w RPA, południowoafrykański przedsiębiorca, pierwszy południowoafrykański astronauta, drugi na świecie kosmiczny turysta [17].



Rysunek 1.3: Linus Benedict Torvalds



Rysunek 1.4: Pingwin Tux

Ukończył studia magisterskie na Uniwersytecie w Kapsztadzie. Prowadził działalność związaną z bezpieczeństwem internetowym i spekulacjami finansowymi, Po osiągnięciu sukcesu materialnego założył firmę Canonical Ltd. z siedzibą na Wyspie Man, producenta systemu operacyjnego Ubuntu.



Rysunek 1.5: Mark Shuttleworth

W środowisku Open Source znany jest z używanego przez siebie pseudonimu sabdfl (ang. Self-Appointed Benevolent Dictator for Life - Samozwańczy dożywotni dobroczynny dyktator). Canonical oprócz wsparcia dla Ubuntu zajmuje się biznesową protekcją innych projektów związanych z wolnym i otwartym oprogramowaniem, takich jak KDE i Gnome.

1.5. Zamiast podsumowania

Czytelniku!

Właśnie wkraczasz w świat ideologii FOSS (ang. Free and Open Source Software - Wolne i otwarte oprogramowanie) [18, 19]. Jeżeli używasz Linuksa od pewnego czasu wiesz już o czym piszę. Jeżeli nie - wkrótce przekonasz się, że wolne i otwarte oprogramowanie może stanowić wyzwanie i przygodę na całe życie. Z systemami typu Unix będziesz utrzymywał kontakt podczas całych studiów. Może już wkrótce przy pomocy wolnego oprogramowania zaczniesz administrować serwerem, może na razie tylko hobbystycznie, później profesjonalnie. Być może Linux zagości w Twoim komputerze osobistym na stałe, stając się domowym centrum informacji i rozrywki.

Po pewnym czasie zauważysz, że należysz już do społeczności świata wolnego i otwartego oprogramowania. Inne systemy operacyjne będą dobre, ale dostrzeżesz, że czegoś im brakuje. Przeprogramujesz swój sposób myślenia w taki sposób, aby wydajniej pracować w wolnych przestrzeniach.

Może się też zdarzyć, że będziesz skazany na pracę w systemach Uniksowych. Nie będziesz używał ich jako zamienników zamkniętego oprogramowania komercyjnego, a z takiego powodu, że nie będzie wersji potrzebnego Ci oprogramowania stworzonej dla Microsoft Windows. Tym większą poczujesz satysfakcję.

Jest prawdopodobne, że za jakiś czas zaczniesz tworzyć swoje własne linuksowe aplikacje.

Być może dzięki Linuksowi zaczniesz zarabiać na życie. Wtedy oprócz wyzwania i przygody - Linux stanie się także Twoim sposobem na Twoje życie i realizację Twoich marzeń.

1.6. Zadania

Zadanie 1

Przeczytaj książkę Sama Williamsa "W obronie wolności (Free as in Freedom)" [20]. Książka w wersji elektronicznej jest dostępna za darmo na stronach wydawnictwa Helion.

Zadanie 2

Zapoznaj się z stronami internetowymi dystrybucji Linuksa, które zostały opisane w tym rozdziale.

Zadanie 3

Postaraj się regularnie odwiedzać portale dotyczące świata Linuksa, np. www.linuxnews.pl.

Zadanie 4

Znajdź na swoim roku grupę studentów zainteresowanych ideologią Linuksa i wolnego oprogramowania.

Zadanie 5

Poproś upoważnione do tego osoby o założenie konta na linuksowym serwerze. Będzie niezbędne do pracy przez cały okres studiów.

Zadanie 6

Jeśli czas i chęci pozwolą zorganizuj na swoim wydziale jakieś przedsięwzięcie związane z promocją idei wolnego oprogramowania.

Rozdział 2

Instalacja i konfiguracja Ubuntu 10.04 $\rm LTS$

| 2.1. | Wprowadzenie | 16 |
|------|-----------------------------|-----------|
| 2.2. | Przygotowanie do instalacji | 16 |
| 2.3. | Proces instalacji | 17 |
| 2.4. | Strojenie systemu | 24 |
| 2.5. | Podsumowanie | 28 |
| 2.6. | Zadania | 29 |

2.1. Wprowadzenie

Do zaprezentowania procesu instalacji systemu operacyjnego wybrano Ubuntu 10.04 LTS Desktop Edition o nazwie kodowej Lucid Lynx.

Zanim przejdziemy do omówienia szczegółów przedstawimy kilka faktów dotyczacych nazewnictwa kolejnych wersji środowiska. Nowa wersja Ubuntu pojawia sie dwa razy w roku, z reguły w kwietniu i w październiku. Zatem numeracja 10.04 informuje nas, że wersja ta została wydana w kwietniu 2010 r. Litery LTS (pojawiające się w Ubuntu co 2 lata) są skrótem od angielskiego Long Term Support oznaczającego długoterminowe wsparcie techniczne - trzy lata dla wersji na biurko (Desktop Edition) i pięć lat dla wersji serwerowej (dla dystrybucji bez LTS mamy gwarantowane wsparcie przez osiemnaście miesięcy). Nazwy kodowe są wyszukanymi nazwami rozmaitych zwierząt, rozpoczynają się od tej samej litery, z wersji na wersje litery te sa kolejnymi literami alfabetu. Wyjatek stanowa wersja 4.10 Warty Warthog i 5.04 Hoary Hedgehog. Następne nazwy budowano już według reguły: 5.10 Breezy Badger, 6.06 Dapper Drake LTS (wydana w czerwcu, pominięto zwierzę na literę C), 6.10 Edgy Eft, 7.04 Feisty Fawn, 7.10 Gutsy Gibbon, 8.04 Hardy Heron LTS, 8.10 Interpid Ibex, 9.04 Jaunty Jackalope, 9.10 Karmic Koala. Wydana w październiku 2010 wersja Ubuntu 10.10 została oznaczona jako Maverick Meerkat.

2.2. Przygotowanie do instalacji

Przygotowując się do instalacji należy bezwzględnie zrobić kopie zapasowe wszystkich danych znajdujących się na dysku komputera, na którym zamierzamy zainstalować Ubuntu. Autorzy skryptu nie ponoszą odpowiedzialności za żadne szkody spowodowane ewentualną utratą danych w wyniku instalacji nowego systemu. Wprawdzie w 99 przypadkach na 100 taka utrata danych nie powinna mieć miejsca, tym nie mniej lepiej jest dmuchać na zimne.

Wymagania sprzętowe systemu Linux są z reguły znacznie mniejsze niż w przypadku Microsoft Windows. Szybko działający system Ubuntu 10.04 LTS z powodzeniem może zostać zainstalowany nawet na komputerze z procesorem Intel Pentium 4 i 512 MB RAM. Znane są instalacje Ubuntu 10.04 LTS na znacznie gorszych platformach sprzętowych.

Należy podkreślić, iż w jednym komputerze można zainstalować więcej niż jeden system operacyjny. Jakkolwiek banalnie by to nie zabrzmiało, chcemy uświadomić nieświadomym, że pliki obu systemów (w przypadku gdy zdecydujemy się zachować Windows) nie będą się wzajemnie mieszać i oba systemy nie powinny wchodzić sobie w drogę (aczkolwiek z poziomu Linux będziemy mieć dostęp do partycji zawierającej system Windows. Cała partycja będzie widziana jako odrębny katalog). Przy starcie komputera przy pomocy prostego menu będzie można wybierać z jakiego systemu chcemy w danej sesji korzystać.

Chociaż Linuksa można zainstalować na znacznie mniejszej partycji, do optymalnej pracy w systemie zaleca się, żeby przeznaczyć na ten cel 20 GB miejsca na twardym dysku. Najlepszym rozwiązaniem w przypadku chęci pozostawienia w komputerze systemu Windows jest zorganizowanie odrębnej partycji, na której zechcemy instalować Ubuntu.

Obrazy z instalacją Ubuntu można ściągnąć ze strony www.ubuntu.pl lub zamówić po założeniu konta w firmie Canonical, całkowicie za darmo, na tłoczonej płycie z okładką. Należy jednak liczyć się z faktem, że na zamówione obrazy czeka się kilka tygodni.

Mimo, że autorzy sympatyzują z polską grupą Ubuntu - na potrzeby tego skryptu i pracy w ramach studiów na kierunkach informatycznych zaleca się korzystanie z oryginalnej wersji dystrybucji systemu, takiej jaką można ściągnąć ze strony www.ubuntu.com/download. Spolszczenie dystrybucji wymaga kilku chwil po instalacji, w oryginalnej wersji Canonical dysponujemy jednak (w opinii autorów) lepiej dobranymi pakietami i zależnościami w porównaniu z podstawową, biurkową wersją polską.

2.3. Proces instalacji

Instalacja systemu w większości przypadków jest prosta i kończy się sukcesem. Wyjątek mogą stanowić szczególnie nietypowe konfiguracje sprzętowe - zwłaszcza karty graficzne, dla których już bądź jeszcze nie ma dostępnych sterowników, zdarza się to jednak niezmiernie rzadko. Można przyjąć, że przy wsparciu społeczności Ubuntu da się zainstalować praktycznie na każdej maszynie. Po uruchomieniu komputera z opcją ładowania systemu operacyjnego z płyty instalacyjnej Ubuntu powinien ukazać się ekran wyboru języka instalacji jak na Rys. 2.1.

Wybieramy zatem język polski i naciskamy Enter.

Po chwili otrzymujemy kolejne menu wyboru (Rys. 2.2). Dla użytkowników kompletnie niezaznajomionych z Ubuntu oraz dla tych, którzy nie są pewni czy system da się zainstalować danej maszynie polecamy wybór opcji "Wypróbuj Ubuntu" bez instalacji. Uruchomimy w ten sposób system w tak zwanej wersji LiveCD. Już wtedy będzie można zaznajomić się z środowiskiem graficznym Gnome, przejrzeć domyślnie zainstalowane aplikacje, a w większości przypadków uda się nawet połączyć z Internetem.

| <u>M</u> aszyna <u>U</u> rządz | zenia Pomo <u>c</u> | 1 | | | | | |
|--------------------------------|---------------------|------------------|-----------------------|----------------|--|--|--|
| | Language | | | | | | |
| Am | haric | Gaeilge | Nederlands | Wolof | | | |
| An | abic | Galego | Norsk bokmål | 中文(简体) | | | |
| As | turianu | Gujarati | Norsk nynorsk | 中文(繁體) | | | |
| Бе | ларуская | עברית | Punjabi(Gurmukhi) | | | | |
| Бъ | лгарски | Hindi | Polski | | | | |
| Be | ngali | Hrvatski | Português do Brasil | | | | |
| Bo | sanski | Magyar | Português | | | | |
| Ca | talà | Bahasa Indonesia | Română | | | | |
| Če | ština | Italiano | Русский | | | | |
| Cy | mraeg | 日本語 | Sámegillii | | | | |
| Da | nsk | ქართული | Slovenčina | | | | |
| De | utsch | Қазақ | Slovenščina | | | | |
| Dz | ongkha | Khmer | Shqip | | | | |
| Eλ | ληνικά | 한국어 | Српски | | | | |
| En | glish | Kurdî | Svenska | | | | |
| Es | peranto | Lietuviškai | Tamil | | | | |
| Es | pañol | Latviski | Thai | | | | |
| Ee | sti | Македонски | Tagalog | | | | |
| Eu | skara | Malayalam | Türkçe | | | | |
| Su | omi | Marathi | Українська | | | | |
| Fri | ançais | Nepali | Tiếng Việt | | | | |
| F1 Help F2 La | nguage F3 k | (eymap F4 Modes | F5 Accessibility F6 O | ther Options | | | |
| | | | 2 🖸 🔁 🕑 | 🛞 💽 Prawy Ctrl | | | |

Rysunek 2.1: Ekran wyboru języka instalacji Ubuntu



Rysunek 2.2: Menu wyboru na początku instalacji systemu

Skoncentrujmy się jednak na procesie instalacji. Wybieramy zatem opcję "Zainstaluj Ubuntu" i naciskamy klawisz Enter. W kolejnym kroku (Rys. 2.3) dokonujemy potwierdzenia wyboru języka polskiego dla instalacji i dokonujemy wyboru strefy czasowej (Rys. 2.4), w której będzie pracował komputer. Tu interfejs instalatora staje się w pełni graficzny, od tej chwili można korzystać z myszy.



Rysunek 2.3: Potwierdzenie wyboru języka

Rys. 2.5 przedstawia kolejne menu instalatora, w którym dokonujemy wyboru układu klawiatury. Domyślnie zalecany jest układ polski, tak zwany układ programisty tym nie mniej w szczególnych przypadkach dysponujemy możliwością wyboru. Warto zwrócić uwagę na okienko wprowadzania tekstu, w którym możemy przetestować każdy wybrany układ.

Najtrudniejszym etapem podczas instalacji Ubuntu jest proces dzielenia dysku i przeznaczenia odpowiednich partycji właściwym modułom systemowym. Można skorzystać z opcji automatycznego dzielenia dysku (z przewodnikiem), zaleca się jednak wybór opcji ustawień ręcznych, dla zaawansowanych (Rys. 2.6). Klikamy przycisk "Dalej" i w kolejnym menu (Rys. 2.7) wybieramy partycję dysku, na którym zainstalujemy Ubuntu. Jeżeli wcześniej w systemie Windows (lub innym) została przygotowana partycja, którą postanowiliśmy wykorzystać dla Linuksa to należy ją w tym miejscu usunąć. Otrzymamy w ten sposób wolny fragment twardego dysku, który zaplanujemy od samego początku. System Linux do poprawnej pracy



Rysunek 2.4: Wybór strefy czasowej

| • | 🕥 Instalacja | |
|---|--|---|
| k | Jkład klawiatury (tóry układ najbardziej przypom Sugerowana opcia: | nina klawiaturę użytkownika? Poland |
| | Wykrywanie mapy klawiatur Własny wybór: | ry Wyknji |
| | Norway Pakistan Poland Portugal Romania | Poland - Dvorak Poland - Dvorak, Polish quotes on ke Poland - Dvorak, Polish quotes on qu Poland - Kashubian |
| 1 | Nożna pisać w tym polu tekstov | wym, aby przetestować nowy układ klawiatury. |
| | Lup 5 2 7 | |

Rysunek 2.5: Wybór układu klawiatury

wymaga tak zwanej partycji wymiany (ang. swap). Stosowana od wielu lat reguła jest przydzielenie na partycję wymiany obszaru dwukrotnie większego od pamięci RAM komputera. Jeżeli zatem w naszym systemie posiadamy 1 GB pamięci RAM, przeznaczmy 2048 MB na przestrzeń wymiany. W tym celu należy kliknąć przycisk "Nowa tablica partycji" i w okienku podobnym do Rys. 2.8 wpisać pożadany rozmiar oraz wybrać opcję "Użyj jako przestrzeń wymiany". Kolejną partycją jaką planujemy utworzyć to partycja główna oznaczana symbolem "/". To na niej będą przechowywane jadro systemu, podstawowe pakiety i inne ważne pliki systemowe. Na początek można przeznaczyć 8 GB miejsca na taką partycję (Rys. 2.8). Wybieramy zatem rozmiar, system plików ext4 i punkt montowania "/". W sposób analogiczny powinniśmy stworzyć również partycję ext4 z punktem montowania /home. Podczas automatycznego podziału dysku nie jest to aranżowane, ale my jako doświadczeniu użytkownicy w sposób szczególny polecamy taki sposób podziału. Na partycji /home beda przechowywane katalogi domowe wszystkich użytkowników systemu. Oznacza to, że na tej wydzielonej partycji będą wszystkie dane zawierające efekty pracy z systemem, dokumenty, grafiki, programy i inne. Gdyby więc zaszła nagła potrzeba reinstalacji systemu - możemy pozostawić partycję /home nieformatowana, przeinstalować sam system i mieć dostęp do starych danych bez konieczności zgrywania ich z kopii zapasowej (która zawsze warto na wszelki wypadek zrobić). To bardzo przyspiesza prace i ułatwia życie. W przypadku instalacji serwerowych inne strategiczne partycje też mogą mieć odrębne punkty montowania co pozwoli uchronić dane w przypadku formatowania partycji głównej. Na partycję /home przeznaczymy całe pozostałe wolne miejsce (instalator sam podpowie ile go jest). Kończąc podział na partycję zatwierdzamy dokonanie zmian i kontynuujemy proces instalacji (Rys. 2.9).

W kilka chwil po rozpoczęciu właściwej instalacji zostaniemy poproszeni o utworzenie pierwszego użytkownika. Pamiętajmy, że pierwszy użytkownik jest jak pierwszy oficer na okręcie. Posiada uprawnienia do wykonywania większości czynności administracyjnych w systemie. Jest szczególnie uprzywilejowany przez co może być niebezpieczny. Jeżeli z komputera korzystasz tylko ty - prawdopodobnie ty będziesz pierwszym oficerem. Jeżeli komputer będzie miał wielu użytkowników - zadbaj o to, żeby pierwszym nie został ktoś niedoświadczony albo nieodpowiedzialny.

Rys. 2.10 przedstawia okienko tworzenia nowego użytkownika. Proszą nas o podanie pełnej nazwy, nazwy skróconej potrzebnej do zalogowania, hasła - od razu otrzymujemy informację zwrotną o jego sile i nazwy komputera, na którym instalujemy Ubuntu. Jeśli w komputerze był wcześniej zainstalowany jakiś system operacyjny (np. Windows) zostaniemy jeszcze poproszeni o potwierdzenie instalacji menedżera ładowania systemu



Rysunek 2.6: Okno wyboru sposobu podziału dysku

| zyna <u>U</u> rządi | zenia Pomo <u>c</u> | | | | | | | |
|---------------------|------------------------|------|------------------|------------------|----------------|---------------|---|------------|
| | | | | | | | | |
| | | | | | | | | |
| | 🔗 Instalacja | | | | | | | |
| | Brzygotowywani | | artveii | | | | | |
| | Przygotowywanie | e pa | агсусјі | | | | | |
| | Free space 8.6 GB | | | | | | - | |
| | Urządzenie | Тур | Punkt montowania | Sformatować? | Rozmiar | Zajęte | 4 | |
| | /dev/sda | | | | | | | |
| | dostępna przestrzeń | | | | 8589 MB | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | • | |
| | Nowa tablica partycji. | | | Usuń Przywró | óć | | | |
| | Etap 5 z 8 | | | Za <u>k</u> ończ | <u>W</u> stecz | <u>D</u> alej | | 3 |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | و 😂 | 7 🗖 🛛 | ۲ | Prawy Ctrl |

Rysunek 2.7: Okno tworzenia nowych partycji
| <u>1</u> aszyna <u>U</u> rządzenia Pomo <u>c</u> | |
|--|-------------------------------|
| | |
| | |
| | |
| 💌 🔿 Instalacja | |
| | |
| | |
| Dodaci zavoji za tanji | C Dadatawaya C Lasiana |
| Rodzaj nowej partycji: | Podstawowa Cogiczna |
| Rozmiar nowej partycji w megabajtach (1000000 bajtów): | 8192 |
| Lokalizacja dla nowej partycji: | Początek Koniec |
| Użyj jako: | Ext4 journaling file system 🔹 |
| Bunkt menteuronia. | |
| Punkt montowania: | |
| | <u>A</u> nuluj <u>O</u> K |
| Nowa tablica partycji Dodaj Zmień U | lsuń Przywróć |
| | |
| Etap 5 z 8 | Zakończ Wstecz Dalej |
| | |
| | |
| | Sector Ctrl |

Rysunek 2.8: Tworzenie partycji podczas instalacji Ubuntu



Rysunek 2.9: Okno zatwierdzenia dokonanych zmian i rozpoczęcia instalacji

Grub najczęściej w pierwszym sektorze pierwszej partycji. Radzimy zgodzić się na to, o co Ubuntu prosi.



Rysunek 2.10: Tworzenie pierwszego użytkownika

Kolejne kilkanaście minut czekamy na kopiowanie plików z płyty i instalację pakietów. Możemy ten czas przeznaczyć na czytanie informacji dostarczanych nam przez formę Canonical, dotyczących możliwości instalowanego systemu. Po zakończeniu instalacji zostaniemy poproszeni o ponowne uruchomienie komputera. Instalator wysunie dysk instalacyjny, po naciśnięciu klawisza Enter komputer uruchomi się ponownie. Teraz dzięki menedżerowi ładowania systemu **Grub** będzie można dokonać wyboru systemu operacyjnego. Po wybraniu Ubuntu 10.04 LTS uruchomi się system operacyjny z graficznym oknem logowania GDM (ang. *Gnome Display Manager*) Rys. 2.11.

Instalacja zakończyła się sukcesem.

2.4. Strojenie systemu

Po zalogowaniu uruchamia się środowisko Gnome z motywem graficznym Ambience. Wiele zależy od tego czy połączenie sieciowe zostało automatycznie nawiązane w procesie instalacji czy też nie. Jeżeli, co zdarza się często, instalator nie zdołał Połączyć się z Internetem - musimy



Rysunek 2.11: Logowanie do systemu - okno GDM

skonfigurować połączenie ręcznie i dokonać niezbędnych aktualizacji systemu. Najprostszą sytuację z podłączaniem Ubuntu do sieci mamy zawsze wtedy gdy dostawca zapewnia automatyczne nadawanie numerów IP na przykład przez usługę DHCP. W warunkach domowych bez trudu nawiążemy też połączenie ze światem w sytuacji, gdy komputer wpięty jest do sieci domowej przez dobrze skonfigurowany router. Instalacje Ubuntu na laptopach z kartami bezprzewodowymi również umożliwiają bezproblemową łączność z domową albo uczelnianą siecią radiową. W pozostałych przypadkach definiujemy połączenie ręcznie korzystając z menu: **System** \rightarrow **Preferencje** \rightarrow **Połączenia sieciowe**. Jeśli natrafimy na poważne kłopoty zawsze można poszukać rady na forach dyskusyjnych, prawie na pewno ktoś już napotkał podobny problem. Zakładamy, że połączenie z Internetem jest pierwszą i najważniejszą czynnością jakiej musi dokonać nowy użytkownik.

Kilka chwil po instalacji systemu na ekranie pojawi się okienko informujące o niekompletności pakietów językowych, w tym przypadku dla języka polskiego (Rys. 2.12). W sytuacji gdy jesteśmy podłączeni do Internetu nie pozostaje nam nic innego niż zezwolenie Ubuntu na pobranie spolszczeń do poszczególnych menu i programów, słowników i tym podobnych. Jeżeli zdecydujemy się na instalację spolszczenia później - możemy zrobić to z menu: System \rightarrow Administration \rightarrow Language Support.



Rysunek 2.12: Propozycja spolszczenia systemu po instalacji

Aktualizacji systemu możemy dokonać w trybie graficznym wykonując program: System \rightarrow Administracja \rightarrow Menedżer aktualizacji. Istnieje jednak bardziej elegancki sposób zarządzania pakietami w Ubuntu. W tym celu wykorzystamy jedno z najpotężniejszych narzędzi informatycznych jakie widział świat: konsolę systemu UNIX. Terminal uruchamiamy wykonując polecenie z menu: Programy \rightarrow Akcesoria \rightarrow Terminal. Dobrze jest mieć zawsze Terminal pod ręką, dlatego warto kliknąć nań prawnym przyciskiem myszy jeszcze w menu i dodać do pulpitu albo panelu. Tysiące pakietów z oprogramowaniem dla Ubuntu znajduje się w Internecie, w tak zwanych repozytoriach. Adresy repozytoriów znajdują się w pliku /etc/apt/sources.list. Domyślnie mamy w nim zgromadzone kilkanaście adresów. To do tego pliku dopisujemy wszystkie inne repozytoriach domyślnych. W celu aktualizacji systemu z linii poleceń musimy najpierw zaktualizować bazę pakietów. Wykonujemy polecenie:

sudo aptitude update

gdzie polecenie **sudo** oznacza ni mniej ni więcej jak tylko wykonaj następne polecenia jako super użytkownik (ang. *super user*) czyli

niejaki **root**. Ciekawostka: nazwa **root** wzięła się od angielskiego słowa oznaczającego korzeń. Wszystkie pliki, katalogi, partycje i urządzenia w systemach UNIX/Linux umieszczone są na drzewie katalogów, którego korzeniem jest właśnie **root** oznaczany i przypominający symbol "/". Potocznie administratorów systemów UNIX nazywano rootami. Bycie rootem stanowi zaszczyt. Zatem po naciśnięciu klawisza Enter zostaniemy poproszeni o podanie **naszego** hasła i system zacznie aktualizować listę pakietów łącząc się po kolei z poszczególnymi repozytoriami. Nie każdy użytkownik może wykonywać polecenia administracyjne korzystając z sudo. Pamiętajmy jednak, że pierwszy użytkownik należy do grupy tzw. sudoersów (lista w pliku /**etc/sudoers**) i dlatego dzięki **sudo** może tyle co prawdziwy **root** Mając zaktualizowaą listę pakietów dokonujemy uaktualnienia systemu wykonując polecenie:

sudo aptitude upgrade

i już nie będziemy proszeni o podawanie hasła do momentu zamknięcia terminala.

Przy pomocy polecenia **aptitude** i podobnego doń polecenia **apt-get** możemy dokonywać instalacji i usuwania oprogramowania z poziomu terminala.

Wykonanie polecenia:

sudo aptitude install mc

albo

sudo apt-get install mc

spowoduje zainstalowanie sympatycznego i niezwykle użytecznego programu Midnight Commander (przypominającego popularnego w dawnych czasach Norton Commandera) w systemie. Możesz wypróbować jego działanie wykonując polecenie:

mc

i wyjść z niego wydając polecenie:

exit

Pakiety usuwamy wykonując powyższe polecenia z opcją **remove** zamiast **install**. Tu jednak występuje różnica między poleceniami **aptitude** i **apt-get**. Pierwsze z nich usuwa zarówno pakiet jak i stowarzyszone z nim biblioteki. Drugie polecenie wszystkie stowarzyszone z pakietem biblioteki pozostawia w systemie. Należy pamiętać, że instalacji oprogramowania można dokonywać również w trybie graficznym: **System** \rightarrow Administracja \rightarrow Menedżer pakietów Synaptic. Kolejnym ważnym krokiem w świeżym systemie Linux będzie ustalenie hasła dla **roota**. Dokonamy tego wykonując polecenie:

sudo passwd

i za pierwszym razem (jeśli zostaniemy o to poproszeni) podajemy własne hasło pierwszego użytkownika, a następnie dwukrotnie nowe hasło dla **roota**. Możemy sprawdzić czy się udało logując się bezpośrednio do konsoli administratora systemu:

su -

gdzie myślnik oznacza wczytanie wszystkich ścieżek dostępu potrzebnych **rootowi** do efektywnej pracy. Szybko opuszczamy konsolę **roota** wykonując polecenie **exit**.

Przygotowując dobre środowisko do pracy powinniśmy zainstalować przeglądarkę internetową Google Chrome:

sudo aptitude install google-chrome-stable

oraz Javę. Dobra i stabilna instalacja Javy wymaga uzupełnienia listy repozytoriów, najlepiej wykonując szereg poleceń:

```
sudo add-apt-repository \
"deb http://archive.canonical.com/ lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-bin \
sun-java6-fonts sun-java6-javadb \
sun-java6-jdk sun-java6-jre sun-java6-plugin
sudo update-alternatives --config java
```

gdzie \ oznacza przełamanie zbyt długiej linii.

Pakiety można też instalować ściągając pliki z rozszerzeniami DEB z Internetu. Podwójne kliknięcie na pakiet spowoduje jego instalację. Na początku jednak radzimy zrezygnować z tego typu działań gdyż mogą w mniejszym lub większym stopniu prowadzić do destabilizacji systemu.

2.5. Podsumowanie

Świadomych informatyków nie trzeba długo namawiać do korzystania z systemu Linux. Zupełnie za darmo otrzymujemy potężne narzędzie sieciowe, przeznaczone zarówno do profesjonalnej pracy programistycznej, graficznej jak i do rozrywki. W podstawowej instalacji otrzymujemy pakiet biurowy Open Office nie ustępujący komercyjnemu oprogramowaniu firmy Microsoft. W dobie e-życia, gdzie coraz więcej spraw załatwia się w przeglądarce internetowej nie ma właściwie znaczenia z jakiego systemu operacyjnego korzystamy. Z drugiej strony wszędzie tam gdzie wymagana jest ekstremalna stabilność dziewięciu na dziesięciu administratorów systemu wybierze Linuksa. Pamiętajmy, że Linux jest zainstalowany na 90% superkomputerów z listy TOP 500. Linux jest szansą zarówno dla państw rozwijających się - gdzie znacznie obniża koszty informatyzacji jak i w państwach rozwiniętych - prowadzi do znaczących oszczędności. Wreszcie Linux to doskonała platforma dydaktyczna pozwalająca niemal na palcach wytłumaczyć funkcjonowanie systemów operacyjnych.

Środowisko zainstalowane tak jak w tym rozdziale pozwala na stabilną pracę z systemem z podstawowymi aplikacjami. Czytelnik został zaznajomiony ponadto ze sposobami instalowania potrzebnych mu w dalszej karierze pakietów. Nie pozostaje zatem nic innego jak tylko życzyć wspaniałej przygody i udanego życia z systemem Ubuntu.

2.6. Zadania

Zadanie 1

Zainstaluj Ubuntu Desktop Edition i podłącz system do Internetu.

Zadanie 2

Zapisz się na forum http://forum.ubuntu.pl.

Zadanie 3

Zainstaluj spolszczenia, Javę i Midnight Commandera w systemie.

Zadanie 4

Zainstaluj program do obróbki grafiki Gimp. Spróbuj otworzyć w nim dowolne zdjęcie, wypróbuj możliwości.

Zadanie 5

Sprawdź jak działa w Twoim systemie pendrive, telefon komórkowy, aparat cyfrowy, skaner, zewnętrzny dysk USB.

Zadanie 6

Spróbuj przez miesiąc korzystać tylko z systemu Linux. Następnie porównaj elastyczność Ubuntu z innymi systemami operacyjnymi.

Rozdział 3

INTERFEJS WIERSZA POLECEŃ

| 3.1. | Podstawy interfejsu wiersza poleceń | | |
|------|-------------------------------------|----------------------------------|-----------|
| | 3.1.1. | Wprowadzenie | 32 |
| | 3.1.2. | Tryb konsolowy | 32 |
| | 3.1.3. | Tryb edycji trybu konsolowego | 33 |
| | 3.1.4. | System pomocy | 34 |
| | 3.1.5. | Zmienne środowiskowe | 35 |
| | 3.1.6. | Cytowania | 36 |
| 3.2. | Strumi | enie standardowe | 38 |
| | 3.2.1. | Deskryptory plików standardowych | 38 |
| | 3.2.2. | Przekierowanie | 38 |
| | 3.2.3. | Urządzenie /dev/null | 39 |
| | 3.2.4. | Potok | 39 |
| | 3.2.5. | Program tee | 40 |
| | | | |

3.1. Podstawy interfejsu wiersza poleceń

3.1.1. Wprowadzenie

Większość obecnych użytkowników komputerów została przyzwyczajona do obsługi komputera z wykorzystaniem graficznego interfejsu użytkownika (*GUI – Graphic User Interface*).

Historycznie starszym jest interfejs w postaci wiersza poleceń, (CLI – Command Line Interface), gdzie odpowiedni program – interpreter, umożliwia nam wpisywanie poleceń z odpowiednimi argumentami. Interfejs ten jest w pierwszym okresie użytkowania trudniejszy, ponieważ w odróżnieniu od GUI wymaga od użytkownika pewnego zasobu wiedzy. Jego możliwości są jednak daleko większe i poznanie takiego sposobu obsługi komputera należy traktować jako obowiązkowe.

Na wspomniane większe możliwości *CLI* wobec *GUI* składa się bogactwo opcji wydawanych poleceń, możliwość zestawiania ich w potoki, a także możliwość wykorzystania w skryptach i tym samym automatyzacji wielu czynności.

W przypadku systemu operacyjnego GNU/Linux zostanie omówiona praca z interpreterem poleceń (tzw. *powłoką*) bash.

Polecenia można podzielić na tzw. *wewnętrzne* (albo *wbudowane*), czyli wykonywane bezpośrednio przez interpreter poleceń oraz na *zewnętrzne*, czyli po prostu programy uruchamiane przez interpreter poleceń. Ponieważ sposób wydawania poleceń w obu przypadkach jest taki sam, zasygnalizowany podział nie będzie wykorzystywany.

3.1.2. Tryb konsolowy

Praca z interpreterem poleceń bywa określana pracą w trybie konsolowym, terminalowym albo tekstowym. Jeżeli używana przez nas dystrybucja systemu GNU/Linux uruchamia się w trybie graficznym, należy w menu wyszukać program o nazwie *Konsola* albo *Terminal*.

W systemie GNU/Linux jest możliwe przełączenie się na rzeczywisty terminal tekstowy. Ilość dostępnych terminali tekstowych zależna jest od konfiguracji systemu i może być ich nawet 6. Przełączanie odbywa się poprzez użycie kombinacji klawiszy Ctrl+Alt+Fn, gdzie Fn oznacza jeden z klawiszy funkcyjnych od F1 do F6. Zazwyczaj siódma konsola jest konsolą graficzną i można ją przywołać kombinacją klawiszy Ctrl+Alt+F7.

Wersję interpretera można sprawdzić poleceniem:

bash --version

Z powłoki bash można wywołać podpowłokę wydając polecenie

bash

Wykonywanie aktualnej powłoki zostaje wówczas zawieszone do czasu zakończenia pracy podpowłoki.

Zakończyć pracę w trybie konsolowym można na kilka sposobów. Można zamknąć okno odpowiedniego programu, wydając jedno z poleceń

exit, logout

a ponadto w oknie terminala można nacisnąć kombinację klawiszy Ctrl+D.

3.1.3. Tryb edycji trybu konsolowego

Okno konsoli posiada własny tryb edycyjny wymagający kilku słów komentarza. Podstawową jednostką edycyjną tego trybu jest pojedynczy wiersz (linia) tekstu. Wydawanie poleceń polega na wpisaniu nazwy polecenia i w razie potrzeby podaniu za nim argumentów wywołania oddzielonych spacjami. Jeżeli argument ma charakter napisu i sam w sobie zawiera spację, wówczas całość napisu należy rozpocząć i zakończyć znakiem cudzysłowu ". Edycję polecenia kończy naciśnięcie klawisza Enter. Przykładowe proste polecenie:

echo "Hello world!"

wyświetli na monitorze dobrze znany napis.

Klawisz Backspace zgodnie z intuicją kasuje poprzedzający znak.

W trakcie edycji można korzystać z klawiszy kursora. Strzałki w lewo, prawo – umożliwiają korektę edytowanego wiersza, strzałki w górę, dół – umożliwiają przeglądanie historii poleceń i wybór albo modyfikację polecenia wydanego wcześniej.

Po wydaniu polecenia

history

zostanie wyświetlona ponumerowana historia wydanych poleceń. Możliwe jest wywołanie polecenia z historii w następujący sposób:

!numer_polecenia

(wykrzyknik numer polecenia w historii).

Po naciśnięciu klawiszy **Ctrl+R** zostanie wywołany tryb przeszukiwania historii. Od tej chwili można zacząć wpisywać dowolny fragment wydanego wcześniej polecenia (niekoniecznie początkowy), a system będzie wyszukiwał, wyświetlał i umożliwiał edycję wydanego wcześniej polecenia, zawierającego wpisany fragment tekstu.

Szczególne znaczenie posiada klawisz tabulatora \mathbf{Tab} , któremu przypisano funkcję autouzupełnienia. Po naciśnięciu klawisza \mathbf{Tab}

program terminala analizuje bieżący kontekst i w sposób inteligentny próbuje uzupełnić wiersz polecenia. W przypadku kontekstu wieloznacznego automatycznie, jest uzupełniany fragment nie budzący wątpliwości, po czym jest konieczne wpisanie fragmentu polecenia eliminującego wieloznaczność.

Możliwe jest wpisanie kilku poleceń oddzielonych znakiem średnika w jednym wierszu i wykonanie ich sekwencyjnie jedno po drugim po naciśnięciu klawisza **Enter**.

3.1.4. System pomocy

Historycznie najstarszym systemem pomocy w systemie GNU/Linux jest tzw. *manual*. Polecenie umożliwiające skorzystanie z tego systemu pomocy powstało przez skrócenie nazwy *manual* do **man**. Podstawowy sposób użycia pokazuje przykład:

man nazwa_polecenia

Trzy najbardziej przydatne opcje polecenia man:

man -a nazwa_polecenia

Opisana wyżej wersja podstawowa pokazuje wyłącznie pierwszą wyszukaną stronę podręcznika dotyczącą nazwa_polecenia. Opcja -a wyszukuje opisy polecenia nazwa_polecenia we wszystkich rozdziałach podręcznika.

man -f nazwa_polecenia

Pokazuje w jakich rozdziałach opisane jest polecenie nazwa_polecenia,

man -k slowo_kluczowe

Wyszukuje w podręczniku strony zawierające w części nagłówkowej slowo_kluczowe.

Możliwe jest również wywołanie:

man n nazwa_polecenia

gdzie n jest numerem interesującego nas rozdziału.

Aby dowiedzieć się więcej o manualu można oczywiście wydać polecenie:

man man

Aby opuścić program umożliwiający przeglądanie stron podręcznika należy nacisnąć klawisz \mathbf{Q} . Inne aktywne klawisze zostaną opisane przy okazji omawiania programów stronicujących.

Wiele poleceń przewiduje użycie argumentów wywołania -h albo --help, wyświetlając wówczas skróconą instrukcję obsługi. Taka pomoc

jest również często wyświetlana w przypadku wywołania z argumentami wadliwymi formalnie.

Alternatywnymi systemami pomocy dla GNU/Linuksa są info i pinfo.

Dla tego systemu operacyjnego kompletną i wygodną w użytkowaniu kopię systemu pomocy (manuala) można znaleźć w internecie [21].

3.1.5. Zmienne środowiskowe

Aktualny zestaw zmiennych środowiskowych powłoki **bash** wyświetlą polecenia

set

printenv

W powłoce **bash** utworzenie nowej zmiennej środowiskowej i nadanie jej wartości odbywa się w następujący sposób:

nazwa_zmiennej=wartosc_zmiennej

Wielkie i małe litery w nazwa_zmiennej są rozróżniane. Sąsiadująco ze znakiem równości = nie może wystąpić spacja. Jeżeli wartosc_zmiennej jest napisem zawierającym spacje, należy całość napisu zamknąć w znakach cudzysłowu.

Zakres obowiązywania tak utworzonej zmiennej to wyłącznie aktualnie używana powłoka **bash**. Aby zmienna była przekazywana do wywoływanych powłok (podpowłok) należy zaznaczyć, że ma być "eksportowana":

export nazwa_zmiennej

Obie instrukcje można wykonać w jednym wierszu:

export nazwa_zmiennej=wartosc_zmiennej

Do zmiennej środowiskowej można się odwołać poprzedzając jej nazwę znakiem \$ (dolara). Na przykład polecenie:

```
echo "$nazwa_zmiennej"
```

wyświetli aktualną wartość zmiennej.

Aby usunąć zmienną można posłużyć się poleceniem:

unset nazwa_zmiennej

Modyfikacje zmiennej przeprowadzone w podpowłoce (włączając w to usunięcie zmiennej) nie będą obowiązywały w powłoce macierzystej.

3.1.6. Cytowania

Interpreter wiersza poleceń (powłoka **bash**) przewiduje trzy rodzaje ograniczników napisów, zwyczajowo nazywane cytowaniami. Są to "cudzysłowy", 'apostrofy' i 'odwrócone apostrofy'.

Zależnie od sposobu cytowania napisy są różnie interpretowane.

Najprostszy przypadek to użycie 'apostrofów'. Napis nimi ograniczony nie jest poddawany żadnej analizie.

Napis ograniczony "cudzysłowami" analizowany jest pod kątem zawierania zmiennych i wyrażeń. Jeżeli takowe zostaną wykryte, to w pierwszej kolejności zostaną zastąpione bieżącymi wartościami zmiennych albo wynikami wykonania wyrażeń.

Napis ograniczony 'odwróconymi apostrofami' zostanie potraktowany w pierwszej kolejności jakby był ograniczony "cudzysłowami" (zostaną podstawione wartości zmiennych oraz wyliczone wyrażenia – jeżeli tylko zmienne i wyrażenia występują), a następnie tak powstały napis zostanie potraktowany jako polecenie. Polecenie zostanie wykonane, a wynik jego działania zastąpi tekst ograniczony 'odwróconymi apostrofami'.

Kilka przykładów:

```
DWA=2
```

```
echo $DWA # 2
echo "$DWA" # 2
echo '$DWA' # $DWA
echo '$DWA' # $DWA
echo '$DWA' #bash: 2: polecenie nieodnalezione
echo $($DWA) #bash: 2: polecenie nieodnalezione
echo $(($DWA+$DWA)) # 4 // bash traktuje $((..)) jako
echo "$(($DWA+$DWA))" # 4 // wyrażenie arytmetyczne
echo '$(($DWA+$DWA))" # 4 // wyrażenie arytmetyczne
echo '$(($DWA+$DWA))' # $(($DWA+$DWA))
echo '$(($DWA+$DWA))' #bash: 4: polecenie nieodnalezione
echo $($(($DWA+$DWA))) #bash: 4: polecenie nieodnalezione
```

W poniższym przykładzie argument +%H:%M określa format (+) w jakim zostanie wyprowadzona obecna data przez program date. %H oznacza godziny, a %M minuty. Pomiędzy nimi został umieszczony znak : (dwukropek).

```
GODZINA="date +%H:%M"
echo $GODZINA #date +%H:%M
echo "$GODZINA" #date +%H:%M
echo '$GODZINA' #$GODZINA
echo '$GODZINA' #16:00
echo $($GODZINA) #16:00
```

Tytułem komentarza należy dodać, że:

- powłoka bash traktuje zapis typu \$((wyrazenie)) jako wyrażenie arytmetyczne i wylicza jego wartość,
- zamiast zapisu 'polecenie' można zastosować zapis \$(polecenie),
- w prostych przypadkach pominięcie znaków cytowania (\$zmienna) jest równoznaczne z cytowaniem w cudzysłowach ("\$zmienna").

Należ unikać uproszczenia z ostatniego punktu. Jeżeli **\$zmienna** zawiera znaki białe, a jest składnikiem większego wyrażenia poddawanego dalszej interpretacji, wówczas uzyskane efekty mogą okazać się niemiłym zaskoczeniem. Prosty przykład:

Jak widać efekt działania trzeciego polecenia **echo** nie musi odpowiadać oczekiwaniom. Należy potraktować jako ćwiczenie wytłumaczenie sobie dlaczego tak się stało.

3.2. Strumienie standardowe

3.2.1. Deskryptory plików standardowych

System operacyjny GNU/Linux w chwili uruchomienia programu (utworzenia procesu) otwiera dla niego trzy standardowe strumienie i przydziela procesowi ich deskryptory:

— standardowe wejście (stdin),

— standardowe wyjście (stdout)

— standardowe wyjście diagnostyczne (błędu) (stderr).

Wartości numeryczne tych deskryptorów to odpowiednio 0, 1 i 2. Używany bywa symbol & (ampersand) oznaczający oba wyjścia równocześnie.

Standardowe wejście jest skojarzone domyślnie z klawiaturą (/dev/stdin). Oba standardowe wyjścia (/dev/stdout, /dev/stderr) są domyślnie skojarzone z monitorem.

3.2.2. Przekierowanie

Bezcenną cechą jest możliwość przekierowania standardowych wejścia, wyjścia i wyjścia diagnostycznego. Jeżeli dowolne polecenie zostanie uruchomione w następujący sposób:

```
polecenie O<plik_We 1>plik_Wy 2>plik_D
```

to zamiast czytać dane wejściowe z klawiatury odczyta je z pliku plik_We. Wyniki normalnie wyświetlane przez program na monitorze zostaną zapisane w pliku plik_Wy, a ewentualne komunikaty diagnostyczne trafią do pliku plik_D.

Powyższy zapis jest równoznaczny z:

```
polecenie <plik_We >plik_Wy 2>plik_D
```

(można pominąć 0 (*zero*) poprzedzające znak < i 1 (*jedynkę*) poprzedzającą znak >). Oczywiście można użyć tylko jednego lub dwóch interesujących nas przekierowań, pozostawiając ustawienia domyślne pozostałych.

Dopuszcza się również przekierowanie:

```
polecenie &>wspolny_plik_wyjsciowy
```

oznaczające, że oba wyjścia zostaną zapisane we wspólnym pliku.

Wszędzie tam, gdzie jest możliwe użycie przekierowania >plik_Wy można również użyć formy:

polecenie >>plik_Wy

W drugim przypadku jeżeli plik_Wy istnieje, wówczas wyniki przekierowania zostaną dołączone do jego końca. W pierwszym przypadku poprzednia zawartość plik_Wy jest tracona i nadpisywana nową treścią.

3.2.3. Urządzenie /dev/null

Zdarza się, że uruchamiany przez nas program oprócz ważnych dla nas wyników produkuje sporo nie interesujących nas informacji diagnostycznych. Można wówczas skorzystać z pliku/urządzenia /dev/null umożliwiającego bezkarne tracenie dowolnie wielu zbędnych informacji:

polecenie 2>/dev/null

3.2.4. Potok

Istnieje bardzo duża grupa programów napisanych tak, by czytały dane ze standardowego wejścia, a wyniki wysyłały na standardowe wyjście. Programy takie nazywane są *filtrami*. System operacyjny zapewnia również możliwość połączenia standardowego wyjścia jednego programu ze standardowym wejściem drugiego. Takie połączenie programów nazywane jest *potokiem*. Składnia połączenia programów w potok jest następująca:

polecenie_1 | polecenie_2

Kluczowy jest oczywiście znak pionowej kreski | (potok, pipe).

Gdyby okazało się, że polecenie_2 musi obrobić wyniki standardowego wyjścia diagnostycznego polecenia_1, wówczas można postąpić następująco (uwaga na nawiasy):

(polecenie_1 >/dev/null) 2>&1 | polecenie_2

Należy zauważyć, że standardowe wyjście tracone jest w /dev/null, a wyjście diagnostyczne zostaje przekierowane na standardowe wyjście 2>&1.

Jeżeli **polecenie_2** ma obrabiać równocześnie wyniki standardowego wyjścia i standardowego wyjścia diagnostycznego (przypadek raczej nietypowy) wówczas składnia jest prostsza:

polecenie_1 2>&1 | polecenie_2

Można oczywiście spowodować, że standardowe wyjście zostanie dołączone do standardowego wyjścia diagnostycznego:

polecenie_1 1>&2

oraz zamienić wyjścia miejscami:

((polecenie_1 1>&3) 2>&1) 3>&2)

chociaż to konstrukcja nietypowa i autorzy nie stanęli wobec konieczności zastosowania jej w rzeczywistości.

3.2.5. Program tee

Wykonując:

```
polecenie_1 | polecenie_2
```

może okazać się, że mimo wszystko zachodzi potrzeba podejrzenia wyników polecenia_1. Umożliwia to program tee. Czyta on dane ze standardowego wejścia, wypisuje je na standardowym wyjściu i zapisuje do wszystkich plików, których nazwy przekazano mu w postaci argumentów wywołania:

```
polecenie_1 | tee plik_podgladu | polecenie_2
```

Rozdział 4

Pliki

| 4.1. | Operacje na plikach | | | |
|------|---------------------|--|-----------|--|
| | 4.1.1. | Wprowadzenie | 42 | |
| | 4.1.2. | Nazwy plików | 42 | |
| | 4.1.3. | Maski nazw plików | 43 | |
| | 4.1.4. | Drzewo katalogów | 44 | |
| | 4.1.5. | Operacje na katalogach | 44 | |
| | 4.1.6. | Operacje na plikach | 46 | |
| 4.2. | Dowiąz | zania do plików | 52 | |
| | 4.2.1. | Wprowadzenie | 52 | |
| | 4.2.2. | Operacje na dowiązaniach | 53 | |
| 4.3. | Atrybuty plików | | | |
| | 4.3.1. | Wprowadzenie | 54 | |
| | 4.3.2. | Operacje na atrybutach plików | 54 | |
| 4.4. | Progra | my do zarządzania plikami | 57 | |
| | 4.4.1. | Wprowadzenie | 57 | |
| | 4.4.2. | Program pilot | 58 | |
| | 4.4.3. | $\operatorname{Program} mc$ - Midnight Commander | 60 | |
| | | | | |

4.1. Operacje na plikach

4.1.1. Wprowadzenie

Z technicznego punktu widzenia można powiedzieć, że praca programisty polega na tworzeniu i przetwarzaniu plików z wykorzystaniem odpowiednich narzędzi. Stąd też niniejszy rozdział zostanie poświęcony omówieniu operacji na plikach.

Osobną kategorię stanowią *pliki tekstowe*. Narzędzia przeznaczone do operowania na plikach tekstowych zostały omówione w osobnym rozdziale.

4.1.2. Nazwy plików

W systemie operacyjnym GNU/Linux nazwy plików i katalogów mogą mieć długość do 256 znaków, chociaż używanie zbyt długich należy uznać za niepraktyczne i prowokujące błędy.

GNU/Linux nie zezwala na użycie w nazwach plików jedynie dwóch znaków: znaku o kodzie ASCII 0 (*zero*) i znaku zarezerwowanego dla separatora katalogów w ścieżce czyli / (ukośnik, *slash*). Z różnych względów (przenośność plików, kłopoty z przekierowaniem, konstruowaniem ścieżki i maski, zdalnym kopiowaniem, pisaniem skryptów itp.) rozsądne jest unikanie w nazwach plików i katalogów następujących znaków:

* (gwiazdka),

```
? (pytajnik),
```

- @ ("małpka"),
- : (dwukropek),
- ; (średnik),
- (minus),
- ~ (tylda),
- / (ukośnik, slash),
- \setminus (odwrócony ukośnik, *backslash*),

```
# (hash),
```

```
^ (daszek, caret),
```

< (znak mniejszości),

```
> (znak większości),
```

```
\mid (potok, pipe),
```

& (ampersand),

```
" (cudzysłów),
```

```
' (apostrof),
```

' (odwrócony apostrof),

- \$ (dolar),
- % (procent),
- () (nawiasy okrągłe),
- [] (nawiasy kwadratowe),
- {} (nawiasy klamrowe)

oraz znaków bez reprezentacji graficznej (w szczególności znaku tabulacji, chociaż spacja też sprawia problemy).

4.1.3. Maski nazw plików

Wykonując różne operacje na plikach często wygodniej jest posłużyć się znakami zastępującymi (wieloznacznikami, *wildcards*) tworząc w ten sposób tzw. maski nazw plików. Dwa znaki mają szczególne znaczenie. Są to ? (pytajnik) i * (gwiazdka). ? (pytajnik) oznacza dokładnie jeden, dowolny znak w nazwie pliku, * (gwiazdka) dowolną liczbę (w szczególności zero) dowolnych znaków. Do maski ??? pasują wszystkie trzyznakowe nazwy plików, a do maski ??* wszystkie nazwy nie krótsze niż 2 znaki. Do maski *.txt pasują wszystkie nazwy plików posiadające na końcu kropkę i litery txt.

W systemie GNU/Linux . (kropka) nie musi oznaczać rozszerzenia pliku – jest dokładnie takim samym znakiem jak każdy inny i może występować w nazwie wielokrotnie i na każdej pozycji. Jeżeli kropka występuje na pierwszej pozycji nazwy pliku, oznacza tzw. plik ukryty.

W systemie GNU/Linux w maskach nazw plików można używać list wartości zamkniętych w nawiasach kwadratowych. Na przykład do maski:

[aeo][dl]a.txt

pasuje nazwa każdego z sześciu plików: ada.txt, ala.txt, eda.txt, ela.txt, ola.txt, ola.txt.

Jeżeli znaki ujęte w nawiasy kwadratowe tworzą ciągły obszar można je wyszczególnić tak jak pokazuje przykład:

[a-z][0-9].png

Do maski tej pasują nazwy, których pierwszym znakiem jest mała litera alfabetu łacińskiego, drugim znakiem jest cyfra, a nazwa kończy się znakami .png. Dopuszczalna jest również konstrukcja:

[a-dp-z]*.txt

oznaczająca pliki z rozszerzeniem .txt, których pierwsza litera zawiera się w przedziałach a-d i p-z.

Należy również wspomnieć, że **bash** rozwija wyrażenia zamknięte w nawiasach klamrowych. Na przykład:

polecenie project/{src,doc,bin/{release,debug}}

jest równoznaczne z serią poleceń:

```
polecenie project/bin/debug
polecenie project/bin/release
polecenie project/doc
polecenie project/src
```

4.1.4. Drzewo katalogów

W każdym linuksowym systemie plików istnieje jeden katalog główny (korzeń, *root*) dla całego drzewa katalogów. W systemach uniksopodobnych oznacza się go znakiem / (ukośnik, *slash*). Ten sam znak (/) pełni funkcję separatora w przypadku wielopoziomowej struktury katalogów. Z taką konwencją jest zgodny zapis:

/pierwszy_poziom/podkatalog/glebszy_podkatalog

Ponadto przez . (*kropkę*) oznacza się katalog bieżący, przez . . (*dwie kropki*) katalog nadrzędny, a znak ~ (*tylda*) oznacza katalog domowy aktualnego użytkownika.

Większość omawianych dalej poleceń charakteryzuje się mnogością możliwych do użycia opcji. Tutaj zostaną omówione tylko najczęściej używane, o największym znaczeniu praktycznym. Z pozostałymi można zapoznać się korzystając z systemu pomocy albo literatury [22].

4.1.5. Operacje na katalogach

pwd (*print working directory*) – wyświetla pełną, bezwzględną nazwę bieżącego katalogu.

cd (*change directory*) – zmienia bieżący katalog, wywołane bez argumentów przechodzi do katalogu domowego bieżącego użytkownika. Przykład:

cd /home

W systemie GNU/Linux istnieją dwie zmienne środowiskowe powiązane z poleceniami pwd i cd. Zmienna **PWD** przechowuje nazwę bieżącego katalogu, a zmienna **OLDPWD** nazwę poprzedniego bieżącego katalogu, czyli nazwę bieżącego katalogu przed wykonaniem ostatniego polecenia cd.

mkdir (make directory) - tworzy nowy katalog lub katalogi. Przykład: mkdir lista nazw katalogow do utworzenia utworzy w bieżącym katalogu 5 katalogów o podanych nazwach.

Polecenie może przyjmować szereg dodatkowych argumentów. Dwa o istotnym znaczeniu praktycznym przedstawiają przykłady:

```
mkdir -m 754 nazwa_katalogu
```

tworzy katalog z odpowiednio ustawionymi prawami dostępu (patrz rozdzia
ł $Atrybuty\ plików),$

mkdir -p nazwa/sciezki/do/utworzenia

tworzy pełną ścieżkę. W przypadku braku któregokolwiek z katalogów rodzicielskich (nadrzędnych) zostanie on również założony. Nie sygnalizuje błędu w przypadku gdy ścieżka już istnieje.

rmdir (*remove directory*) – usuwa katalog lub katalogi o ile są puste, na przykład:

rmdir lista nazw katalogow do usuniecia

usunie z bieżącego katalogu 5 katalogów o podanych nazwach.

Podobnie do mkdir można użyć wariantu:

rmdir -p nazwa/sciezki/do/usuniecia

co jest odpowiednikiem czterech poleceń **rmdir** wydanych w następującej kolejności:

```
rmdir nazwa/sciezki/do/usuniecia
rmdir nazwa/sciezki/do
rmdir nazwa/sciezki
rmdir nazwa
```

Pewną niedogodnością polecenia **rmdir** jest usuwanie wyłącznie pustych katalogów. Usunięcie katalogu łącznie z zawartością umożliwia polecenie:

```
rm -r nazwa_katalogu
```

Polecenie rm będzie dokładniej omawiane przy operacjach na plikach.

basename – podaje część ścieżki za ostatnim znakiem /. Przykład:

```
basename /sciezka/do/pliku
```

wyświetli pliku.

```
dirname – podaje część ścieżki przed ostatnim znakiem /. Przykład:
```

```
dirname /sciezka/do/pliku
```

wyświetli /sciezka/do.

W typowych dystrybucjach GNU/Linuksa stosunkowo rzadko spotykanym programem jest **tree**. Program umożliwia wyświetlenie drzewa katalogów (również z plikami) w postaci semigraficznej. Podstawowa forma wywołania:

tree [opcje] [katalog]

Domyślnie program tree rozpoczyna listowanie od bieżącego katalogu łącznie z plikami. Aby ograniczyć listowanie wyłącznie do nazw katalogów należy użyć opcji –d. Można również określić katalog, od którego tree ma rozpocząć się listowanie.

4.1.6. Operacje na plikach

 ${\tt ls}~(list)$ – wyświetla zawartość katalogu. Polecenie wywołane bez argumentów listuje zawartość bieżącego katalogu w formie krótkiej (wyłącznie nazwy plików) posortowanej alfabetycznie. Forma ogólna wywołania ma postać:

ls [opcje] [plik]

przy czym plik należy rozumieć jako nazwę pliku, listę nazw albo maskę.

Przydatne opcje:

- -a wyświetla wszystkie pliki również te, których nazwy rozpoczynają się od kropki, czyli traktowane jako ukryte,
- -A jak wyżej, ale pomija . (kropkę) i .. (dwie kropki),
- -c z –1 podaje czas ostatniej modyfikacji pliku,
- -d normalnie polecenie ls -l katalog wyświetla zawartość katalogu w formie długiej, polecenie ls -ld katalog wyświetli informacje o katalogu (uprawnienia, właściciel, grupa itd.) tak jak ma to miejsce dla pliku,
- -i dodatkowo wyświetli numery węzłów (i-node),
- -1 forma długa (informacje o uprawnieniach, właścicielu, grupie, rozmiarze i czasie ostatniej modyfikacji pliku,
- -R rekursywnie wyświetla również zawartość wszystkich podkatalogów,
- $-\mathbf{u}$ z $-\mathbf{l}$ podaje czas utworzenia pliku.

Listowanie katalogu w formie długiej $(\tt ls -l)$ powoduje uzyskanie wyniku w następującej postaci:

-rwxr-xr-- l wlasciciel grupa rozmiar czas plik

Pierwszy znak (w przykładzie -) oznacza typ pliku i może mieć postać:

- - zwykły plik,
- ${\tt d} katalog,$
- 1 dowiązanie symboliczne,
- s gniazdo,
- ${\tt p} {\rm potok},$
- c urządzenie znakowe,
- b urządzenie blokowe.

Za typem pliku podawane są uprawnienia (omówione przy poleceniu chmod), 1 oznacza licznik dowiązań twardych (dowiązania jako takie zostaną omówione przy poleceniu ln), dalej przedstawiony jest właściciel, grupa i rozmiar pliku. czas może opcjonalnie oznaczać czas modyfikacji (opcja -c) albo czas ostatniego dostępu (opcja -u). plik oznacza nazwę pliku.

find – program służy do poszukiwania plików znajdujących się w nieokreślonym miejscu struktury katalogów. Program rozpoczyna wyszukiwanie od bieżącego katalogu, ale można zadać argumentem wywołania inny punkt startowy. Kryteria, jakimi można określić pliki poszukiwane przez ten program, obejmują między innymi rozmiar, uprawnienia, czas modyfikacji, właściciela, grupę, ale najczęstszą potrzebą jest wyszukanie pliku o zadanej nazwie lub o nazwie pasującej do maski.

Przykłady:

find – wyświetli wszystkie pliki poczynając od bieżącego katalogu,

find / -name *.c – przeszuka cały system plików w poszukiwaniu tekstów źródłowych programów w języku C,

find /lib -lname libc-2.7.so – w katalogu /lib i jego podkatalogach wyszuka te dowiązania symboliczne, które wskazują na plik libc-2.7.so,

find /usr/bin -size +1000k -size -2000k – w katalogu /usr/bin i jego podkatalogach wyszuka pliki o rozmiarze od 1 do 2 megabajtów,

find /var -user student - w katalogu /var wyszuka piki, których właścicielem jest student,

find -mmin -60 – w bieżącym katalogu i podkatalogach wyszuka pliki zmodyfikowane w ciągu ostatniej godziny.

 ${\sf cp}\;(copy)$ – kopiuje pliki i katalogi. Forma ogólna wywołania ma postać:

cp [opcje] [plik_zrodlowy] [plik_docelowy]

plik_zrodlowy oznaczać może plik, listę plików, katalog albo maskę. plik_docelowy oznacza plik tylko i wyłącznie wówczas, gdy plik_zrodlowy

jest pojedynczym plikiem i nie ma katalogu o nazwie plik_docelowy. Jeżeli plik_zrodlowy jest listą albo maską, do której pasuje nazwa więcej niż jednego pliku, wówczas plik_docelowy oznacza katalog. Jeżeli plik_zrodlowy jest katalogiem, wówczas jest konieczne użycie opcji -r (albo -R) i plik_docelowy oznacza katalog, który musi istnieć. Jeżeli plik_docelowy istnieje, wówczas zostaje zastąpiony nową kopią.

Przydatne opcje:

- -a (archive) tworzone jest archiwum, czyli kopiowana jest struktura drzewa katalogów z pełną zawartością (łącznie z plikami ukrytymi, dowiązaniami itd.), pliki kopii mają ustawiane atrybuty zgodne z atrybutami oryginału,
- -b (backup) w przypadku gdy istnieje plik_docelowy, wówczas w pierwszej kolejności zmieniana jest jego nazwa na plik_docelowy~, a kopiowanie następuje w drugiej kolejności,
- -f (force) w przypadku gdy wystąpił błąd kopiowania i istnieje plik_docelowy, wówczas podejmuje próbę usunięcia plik_docelowy i ponawia próbę kopiowania,
- -i (*interactive*) w przypadku gdy istnieje plik_docelowy, wówczas pyta o pozwolenie zastąpienia,
- -p (preserve) o ile to możliwe zostaną również skopiowane atrybuty pliku takie jak uprawnienia, właściciel, grupa oraz czas ostatniej modyfikacji bez tej opcji kopia ma ustawionego właściciela na użytkownika kopiującego, a nie właściciela oryginału, natomiast czas modyfikacji jest ustawiany na czas kopiowania, a nie na czas modyfikacji oryginału,
- -r (-R) (recursive) kopiuje rekursywnie podkatalogi,
- -u (update) kopiuje tylko takie pliki, dla których plik_zrodlowy jest młodszy niż plik_docelowy albo nie ma pliku plik_docelowy.

Dwie ostanie opcje dublują omawiane dalej polecenie ln:

- -1~(link)– zamiast kopiować tworzy dowiązanie twarde do pliku,
- $-{\bf s}~(symbolic~link)$ zamiast kopiować tworzy dowiązanie symboliczne do pliku.

 ${\tt mv}~(move)$ – zmienia nazwę i/lub przenosi pliki pomiędzy katalogami. Forma ogólna wywołania ma postać:

```
mv [opcje] [plik_zrodlowy] [plik_docelowy]
```

a znaczenie argumentów jest analogiczne do argumentów polecenia cp.

Przydatne opcje:

- -b (backup) w przypadku gdy istnieje plik_docelowy, wówczas w pierwszej kolejności jego nazwa jest zmieniana na plik_docelowy~, a następnie jest zmieniana nazwa plik_zrodlowy na plik_docelowy,
- -f (force) nie pyta o pozwolenie w przypadku gdy plik_docelowy istnieje
 i zostanie zastąpiony plikiem plik_zrodlowy,
- -i (*interactive*) w przypadku gdy istnieje plik_docelowy, wówczas pyta o pozwolenie zastąpienia,
- -u (update) przenosi tylko takie pliki, dla których plik_zrodlowy jest młodszy niż plik_docelowy, albo nie ma pliku plik_docelowy.

 ${\tt rm}~(remove)$ – usuwa pliki lub katalogi. Forma ogólna wywołania ma postać:

```
rm [opcje] [plik]
```

gdzie plik oznacza plik, katalog, listę albo maskę.

Przydatne opcje:

- -d (*directory*) usuwa katalog nawet jeżeli nie jest pusty (opcja działa tylko w trybie administratora, nie wszystkie systemy zapewniają odpowiednie wsparcie),
- -f (force) wymusza usunięcie plików, nie sygnalizuje błędów,
- -i (interactive) oczekuje potwierdzenia przed usunięciem,
- -r (-R) (recursive) usuwa rekursywnie podkatalogi razem z zawartością.

tar (*tape archive*) – tworzy archiwa czyli pliki, w których zapisano wiele innych plików. Nazwa pochodzi z czasów kiedy archiwa takie tworzono na napędach taśmowych. W systemie operacyjnym GNU/Linux archiwa tworzone programem tar są bardzo powszechną formą wymiany plików. Forma ogólna wywołania ma postać:

tar [operacja][opcje]

Wymienienione zostaną tylko ważniejsze operacje, które można określić jedną z liter:

- $-\mathbf{c}$ utwórz nowy plik archiwum, jeżeli plik istnieje zostanie zastąpiony nowym,
- $-\mathbf{r}$ dołącz pliki do istniejącego archiwum,
- -t listuj zawartość archiwum,
- $-{\tt u}$ aktualizuj dołącza do archiwum tylko te pliki, które są nowsze od już w nim zapisanych, starsze pliki pozostają w archiwum nie są zastępowane,
- -x– odzyskuje pliki z archiwum.

Operacji dołączania i aktualizacji nie można przeprowadzić na archiwach skompresowanych.

Najczęściej używane opcje:

-j – stosuje kompresję w formacie bzip2,

 $-{\bf p}$ – zapisuje również atrybuty plików (atrybuty zostały omówione w jednym z następnych rozdziałów),

 $-\mathbf{v}$ – wypisuje nazwy aktualnie przetwarzanych plików,

-z – stosuje kompresję w formacie gzip,

 $\operatorname{\mathsf{--wildcards}}$ – umożliwia stosowanie wieloznaczników przy określaniu plików.

Wymienione opcje stosowane są w połączeniu z opcją $-\mathtt{f},$ która poprzedza nazwę pliku archiwum.

Przykłady:

tar -cvzf ch07.tar.gz ch07/ – utworzy skompresowane formatem gzip archiwum o nazwie ch07.tar.gz zawierające wszystkie pliki z katalogu ch07. Podczas tworzenia archiwum będzie wyświetlał nazwy przetwarzanych plików,

tar -xzf ch07.tar.gz - odtworzy wszystkie pliki ze skompresowanego
formatem gzip archiwum o nazwie ch07.tar.gz,

tar --wildcards -xzf ch07.tar.gz ch07/*.txt – odtworzy ze skompresowanego formatem gzip archiwum o nazwie ch07.tar.gz wszystkie pliki, których nazwa pasuje do maski *.txt, pochodzące z katalogu ch07,

 $\verb+tar-tzf ch07.tar.gz - wylistuje zawartość skompresowanego formatem gzip archiwum o nazwie ch07.tar.gz,$

tar -xzf ch07.tar.gz ch07/gnome_cp.png ch07/gnome_cp.eps - odtworzy ze skompresowanego formatem gzip archiwum o nazwie ch07.tar.gz pliki ch07/gnome_cp.png i ch07/gnome_cp.eps.

 ${\tt zip}$ – tworzy archiwa w formacie zip. Forma ogólna wywołania:

zip [opcje] nazwa_archiwum lista_plikow

Użyteczne opcje:

- -d– usuń pliki z archiwum,
- -i maska w tworzonym archiwum uwzględni wyłącznie pliki o nazwach pasujących do maski stosowane z $-\mathbf{r},$
- -r katalog przetworzy rekursywnie podkatalogi zaczynając od katalog,
- -R przetworzy rekursywnie podkatalogi zaczynając od bieżącego,
- $-{\tt m}$ po umieszczeniu w archiwum pliki źródłowe zostaną skasowane.

Przykłady:

zip txt.zip *.txt – utworzy archiwum o nazwie txt.zip zawierające wszystkie pliki z bieżącego katalogu, których nazwa kończy się na .txt, zip -R all.zip '*' – utworzy archiwum o nazwie all.zip zawierające wszystkie pliki z bieżącego katalogu i z jego podkatalogów, zip c.zip -r src -i '*.c' – utworzy archiwum o nazwie c.zip zawierające wszystkie pliki źródłowe programów w języku C, rekursywnie – poczynając od katalogu o nazwie src.

 ${\tt unzip}$ – program odzyskuje pliki zapisane w archiwach zip. Forma ogólna wywołania:

unzip [opcje] nazwa_archiwum [lista_plikow]

Użyteczne opcje:

-j – odzyskuje pliki do bieżącego katalogu, nie honoruje struktury katalogów zapisanej w archiwum,

-t – testuje czy plik archiwum nie został uszkodzony.

Przykłady:

unzip archiwum.zip – odzyskuje wszystkie pliki z archiwum.zip odtwarzając strukturę katalogów,

unzip -j archiwum.zip '*.txt' – odzyskuje z archiwum.zip pliki o nazwach kończących się .txt i zapisuje je w bieżącym katalogu,

4.2. Dowiązania do plików

4.2.1. Wprowadzenie

W systemie GNU/Linux istnieją dwa rodzaje dowiązań do plików, tak zwane dowiązania twarde i symboliczne.

Dowiązanie twarde jest dodatkowym wpisem w katalogu odwołującym się do tego samego fizycznego obszaru na dysku (i-węzła). Mówiąc innymi słowy do jednego pliku można odwoływać się używając różnych, ale w pełni równoprawnych nazw. Po utworzeniu dowiązania twardego użytkownik nie ma możliwości odróżnienia, który wpis był pierwszy, a który został utworzony później jako dowiązanie. System operacyjny zlicza i na bieżąco obsługuje liczbę dowiązań twardych do pliku. Usunięcie jednego z dowiązań twardych usuwa wpis w katalogu i zmniejsza wskazanie licznika dowiązań. Sam plik jest usuwany tylko w sytuacji, gdy zostanie wyzerowana liczba dowiązań twardych do niego.

Dowiązanie symboliczne jest plikiem przechowującym lokalizację (nazwę pliku w razie potrzeby również z nazwą katalogu) innego pliku. Wszelkie próby dostępu do dowiązania symbolicznego są przekierowywane do wskazywanego przez nie pliku. Usunięcie dowiązania symbolicznego usuwa plik przekierowujący i nie oddziałuje w żaden sposób na plik wskazywany. Uprawnienia ustawione dla pliku dowiązania symbolicznego są ignorowane i znaczenie mają tylko uprawnienia pliku wskazywanego.

Teoretycznie w systemach operacyjnych zgodnych z Uniksem (w tym GNU/Linux) nie ogranicza się możliwości tworzenia dowiązań twardych do katalogów. Co prawda z dokumentacji wynika, że taką możliwość posiada wyłącznie *superużytkownik* (root), jednak w rzeczywistości funkcja taka nie została chyba zaimplementowana w żadnym z popularnych systemów plików. Bezpieczniej jest przyjąć, że do katalogów można tworzyć wyłącznie dowiązania symboliczne.

Do czego używane są dowiązania? Szczególnie dużo dowiązań znajduje się w katalogu /lib. Jako przykład może posłużyć dowiązanie symboliczne libm.so.6 wskazujące (na przykład) na bibliotekę libm-2.10.1.so. Na pewnym etapie rozwoju systemu istniała faktycznie biblioteka o nazwie libm.so.6, jednak później jej funkcję przejęła nowsza wersja o nazwie libm-2.10.1.so. W systemie mogą istnieć starsze wersje oprogramowania odwołujące się nadal do nazwy libm.so.6. Zamiast przechowywać dwie (albo więcej) wersje biblioteki można posługiwać się wyłącznie wersją najbardziej rozwiniętą, a na potrzeby programów, być może odwołujących się do starszej wersji, utworzyć dowiązania gwarantujące poprawne działanie tych programów.

4.2.2. Operacje na dowiązaniach

Omawiane wcześniej polecenie **cp** użyte z odpowiednimi opcjami (-1, -s) potrafi utworzyć oba rodzaje dowiązań, jednak podstawowym poleceniem przeznaczonym do ich tworzenia jest polecenie $\ln (link)$, które w przypadku użycia opcji -s tworzy dowiązanie symboliczne, a w przypadku pominięcia tej opcji dowiązanie twarde.

Poniżej omówione zostaną cztery warianty polecenia ln:

ln [opcje] cel dowiazanie - tworzy dowiązanie o nazwie
dowiazanie do pliku o nazwie cel,

ln [opcje] katalog/cel - tworzy w bieżącym katalogu dowiązanie
o nazwie cel do pliku cel znajdującego się w katalogu katalog,

ln [opcje] cel katalog - tworzy dowiązanie do pliku cel o nazwie cel w katalogu katalog,

ln [opcje] -t katalog cel - tworzy w katalogu katalog dowiązanie do pliku cel o nazwie cel (inna składnia poprzedniego polecenia).

W wersji trzeciej i czwartej polecenia **cel** może być listą nazw plików albo maską nazw plików. Wówczas we wskazanym katalogu zostaną utworzone dowiązania do wszystkich plików listy albo plików, których nazwy pasują do maski. Nazwy dowiązań będą oczywiście zgodne z nazwami linkowanych plików.

Przydatne opcje:

- -b jeżeli istnieje dowiazanie to jego nazwa zostanie zmieniona na dowiazanie[~], a żądane dowiazanie zostanie utworzone w drugiej kolejności,
- -f jeżeli istnieje dowiazanie to zostanie usunięte i zastąpione nowym,
- -n zabezpiecza przed przekierowaniem w przypadku istniejącego dowiązania (szczegóły niżej),
- -s tworzy dowiązanie symboliczne,
- $-\mathtt{t}$ tak jak w czwartej wersji wskazuje katalog docelowy,
- -T wymusza traktowanie dowiązania jako pliku a nie katalogu (szczegóły niżej).

Najważniejszą opcją polecenia jest oczywiście -s, jej użycie decyduje o tworzeniu dowiązań symbolicznych, a pominięcie o tworzeniu dowiązań twardych. Jeżeli plik dowiazanie już istnieje, wówczas polecenie ln przerywa pracę sygnalizując błąd. Dwie pierwsze z omówionych niżej opcji zmieniają to zachowanie:

-T – ta opcja wymaga dłuższego omówienia. Przyjmijmy, że należy wykonać polecenie ln cel dowiazanie (tak jak w pierwszym wariancie) ale w bieżącym katalogu istnieje katalog o nazwie **dowiazanie**. Polecenie zostanie więc wykonane niezgodnie z intencjami wg wariantu trzeciego. Użycie opcji –T zabezpiecza nas przed taką sytuacją i gwarantuje, że **dowiazanie** oznacza nazwę pliku, a jeżeli istnieje katalog o kolidującej nazwie, wówczas polecenie zakończy pracę sygnalizując błąd.

-n – ta opcja również wymaga dłuższego omówienia. Jak wspomniano wcześniej do katalogów można tworzyć wyłącznie dowiązania symboliczne. Załóżmy, że istnieją dwa katalogi o nazwach dir1 i dir2 oraz dowiązanie symboliczne dir wskazujące na dir1. Celem jest, by dir przestał wskazywać dir1, a zaczął wskazywać dir2. Najprostsza metoda to usunięcie dowiązania dir (rm dir) i utworzenie go na nowo, tak by wskazywało tym razem na dir2 (ln -s dir2 dir). Znając jednak opcję -f, wymuszającą zastąpienie istniejącego dowiązania nowym można się spodziewać, że polecenie ln -sf dir2 dir wykona całą pracę w jednym kroku. Niestety dir jest dowiązaniem do katalogu, czyli polecenie zostanie wykonane wg wariantu trzeciego i ostatecznie w katalogu dir1 (wskazywanym przez dir) zostanie utworzone dowiązanie symboliczne do katalogu dir2, co nie jest zgodne z oczekiwanym rezultatem.

Problem tego typu można rozwiązać używając opcji –n. Jeżeli polecenie ln (z opcjami –sf) odnosi się do istniejącego dowiązania symbolicznego do katalogu, to użycie opcji –n wymusi potraktowanie dowiązania jako pliku (nie zostanie wykonane przekierowanie).

Podsumowując: jeżeli dir jest dowiązaniem symbolicznym do katalogu dir1 i należy przestawić to dowiązanie na katalog dir2, to można wykonać polecenie:

ln -sfn dir2 dir.

Niektóre implementacje polecenia ln dopuszczają użycie opcji -h, o dokładnie takim samym znaczeniu (działaniu) jak opcja -n.

4.3. Atrybuty plików

4.3.1. Wprowadzenie

W systemie GNU/Linux przez atrybuty pliku należy rozumieć jego właściciela, grupę, uprawnienia dostępu do pliku oraz czasy ostatniego dostępu i modyfikacji pliku. W niniejszym rozdziale zostaną omówione polecenia umożliwiające modyfikację tych atrybutów.

4.3.2. Operacje na atrybutach plików

chown (*change owner*) – zmienia właściciela i/lub grupę pliku. Polecenie można wywołać na dwa główne sposoby:

```
chown [opcje] [nowy_wlasciciel][:[nowa_grupa]] plik...
chown [opcje] --reference=plik_wzorcowy plik...
```

plik... oznacza plik, listę plików albo maskę, do której mają pasować nazwy modyfikowanych plików.

Pierwszy sposób pokazuje możliwość zmiany tylko właściciela, tylko grupy albo obu atrybutów równocześnie (grupę należy poprzedzić znakiem dwukropka).

Drugi sposób pokazuje jak można wskazać plik, z którego zostaną pobrane atrybuty zastosowane następnie do modyfikowanego pliku....

Jedną z bardziej przydatnych opcji jest: $-{\tt R}$ – dokonaj zmian rekurencyjnie w podkatalogach.

Polecenie to wymaga uprawnień administratora. W przeciwnym wypadku efektywna zmiana właściciela pliku nie będzie możliwa.

chgrp (*change group*) – umożliwia wyłącznie zmianę grupy pliku. Jest bliźniaczo podobne do omówionego wyżej **chown** co do opcji, znaczenia argumentów i ograniczeń zastosowania. Przykłady:

```
chgrp [opcje] nowa_grupa plik...
chgrp [opcje] --reference=plik_wzorcowy plik...
```

chmod (change mode) – umożliwia zmianę uprawnień dostępu do pliku.

Z punktu widzenia uprawnień dostępu do pliku wszyscy użytkownicy podzieleni są na trzy kategorie: właściciel (u), grupa (g) i pozostali (o). Aby ułatwić zmianę uprawnień utworzono również grupę wszyscy (a=u+g+o).

Każda z kategorii użytkowników może mieć (albo nie) prawo do odczytu (\mathbf{r}) , zapisu (\mathbf{w}) i wykonywania pliku (\mathbf{x}) . Prawo do zapisu jest równoznaczne z prawem do usunięcia (skasowania) pliku.

Jeżeli plik jest katalogiem, prawo do wykonywania (x) oznacza prawo dostępu do katalogu i plików zawartych w katalogu – bez tego uprawnienia nie można wykonać żadnych operacji na katalogu ani na plikach w nim zawartych. Bez prawa odczytu (r) katalogu, nie jest możliwe wykonanie operacji listowania (1s) katalogu, ale można wykonać operację zmiany katalogu (cd katalog). W przypadku znajomości nazw plików znajdujących się w tym katalogu, jest możliwe wykonywanie na nich tych operacji, na które zezwalają atrybuty plików. Jest możliwe w takiej sytuacji posługiwanie się nazwami typu katalog/plik.

Podczas omawiania polecenia 1s pokazano, że listowanie katalogu w formie długiej (1s -1) wyprowadza wynik następującej postaci:

-rwxr-xr-- l wlasciciel grupa rozmiar czas plik

W przedstawionym przykładzie uprawnienia wlasciciela do pliku określa trójka rwx, uprawnienia grupy trójka r-x, natomiast uprawnienia wszystkich pozostałych trójka r--.

Zmiany uprawnień można dokonać na 3 sposoby:

```
chmod [opcje] tryb[,tryb]... plik...
chmod [opcje] tryb_oktalnie plik...
chmod [opcje] --reference=plik_wzorcowy plik...
```

plik... oznacza plik, listę plików albo maskę.

Znacznie łatwiej zrozumieć powyższe zapisy na przykładach:

chmod u=rwx,g=rx,o=r plik – ustawia uprawnienia zgodnie z przykładem wyżej.

chmod u-w,o+x plik – zabiera właścicielowi uprawnienia do zapisu, a wszystkim pozostałym daje uprawnienia wykonywania,

chmod a+w plik – nadaje zarówno właścicielowi, grupie jak i wszystkim pozostałym uprawnienia do modyfikacji (zapisu) pliku. To polecenie jest równoważne z poleceniem:

chmod ugo+w plik.

Każda trójka rw
x może być traktowana jako cyfra ósemkowa, gdzie x ma wag
ę 1, w wagę 2, a r wagę 4. Tak więc zamiast:

chmod u=rwx,g=rx,o=r plik można wydać polecenie

chmod 754 plik

Ostatni wariant:

```
chmod [opcje] --reference=plik_wzorcowy plik...
```

można zrozumieć przez analogię do poleceń chown i chgrp. plik będzie miał nadane uprawnienia takie same jak plik_wzorcowy.

Podobnie do poleceń chown i chgrp jedną z bardziej użytecznych opcji jest -R – użycie jej spowoduje zmianę uprawnień rekursywnie we wszystkich podkatalogach.

Ostatnim omawianym poleceniem służącym do modyfikacji atrybutów plików będzie polecenie **touch**. Służy ono do modyfikacji czasu określającego ostatni dostęp i/lub modyfikację pliku. Forma ogólna wywołania:

```
touch [opcje] plik...
```

plik... oznacza plik, listę plików albo maskę nazw plików. Polecenie użyte bez żadnych opcji ustawia zarówno czas modyfikacji jak i ostatniego dostępu na czas aktualnie wskazywany przez zegar systemowy komputera.

Użyteczne opcje:

- -a powoduje ustawienia tylko czasu ostatniego dostępu,
- -m powoduje ustawienia tylko czasu modyfikacji,
- -t CCYYMMDDhhmm.ss narzuca czas zadany argumentem zamiast czasu wskazywanego przez zegar systemowy.
 - \mathtt{CC} oznacza stulecie,
 - YY ostatnie dwie cyfry roku,
 - MM miesiąc,
 - DD dzień,
 - hh godzinę,
 - mm minute,

 $\tt.ss$ – sekundę narzucanego czasu.

Można również posłużyć się jednym z wariantów opcji -d:

```
-d CCYY-MM-DD,
```

- -d hh[:mm[:ss]] minuty i sekundy są opcjonalne,
- -d "CCYY-MM-DD hh[:mm[:ss]]".

Format daty i godziny jest zgodny z formatem w jakim dane te wyświetla polecenie 1s –1. Jest możliwe wskazanie pliku, który ma być wzorcem czasu modyfikacji i dostępu tak jak to pokazuje przykład:

```
touch -r plik_wzorcowy plik...
```

W przypadku gdy plik nie istnieje, dodatkowym efektem działania polecenia touch jest założenie pustego pliku. Cecha ta może okazać się użyteczna, ale może być także niepożądana. Aby uniknąć przypadkowego utworzenia pliku należy posłużyć się opcją -c (*do not create*).

4.4. Programy do zarządzania plikami

4.4.1. Wprowadzenie

Trzy poprzednie rozdziały zostały poświęcone plikom. Przekazano tam więcej niż podstawowy zakres wiedzy i opisano bogaty zestaw poleceń umożliwiających wszechstronne operacje na plikach. Pod względem potrzeb programisty ten zasób wiedzy i umiejętności jest kompletny, a jedynym zastrzeżeniem jakie można wobec niego sformułować jest brak wygody i wynikająca z niego niezbyt wysoka efektywność operacji na plikach. W takiej sytuacji naturalne wydaje się, że dla ułatwienia pracy sobie (i nie tylko) programiści opracowali narzędzia ułatwiające zarządzanie plikami.

Programy do zarządzania plikami (zarządcy plików) pracujące w trybie graficznym opierają się na podobnych wzorcach i znając jeden z nich można łatwo nauczyć się obsługi innych.

Nieco trudniej jest w przypadku narzędzi pracujących w trybie tekstowym. Poniżej zostaną omówione dwa z nich: pilot i mc (Midnight Commander).

4.4.2. Program pilot

pilot to jeden z trójki programów pine, pico, pilot, które powstały na Uniwersytecie Waszyngtońskim z zastrzeżeniem komercyjnego wykorzystania ich nazw. Stąd też pilot nie znajduje się we wszystkich dystrybucjach GNU/Linuksa.

pico jest edytorem tekstu, który doczekał się zamiennika nano dostępnego na licencji GPL. Edytor nano został opisany w rozdziale poświęconym podstawowym edytorom tekstu.

pine jest programem do obsługi poczty elektronicznej w trybie tekstowym. Jego zamiennikiem dostępnym na zasadach licencji Apache jest program alpine.

pilot jest bardzo prostym w obsłudze, wyposażonym tylko w podstawowe funkcje zarządcą plików. Jego największą zaletą jest rozmiar – znacznie poniżej 1 MB.

Program pilot honoruje ustawienia zmiennych środowiskowych PAGER i EDITOR. Jeżeli jednak ich nie znajdzie, do podglądania zawartości i edycji plików wykorzystuje programy pine i pico. Oba programy bez trudu można zastąpić linkami symbolicznymi pico → nano, pine → alpine, zdecydowanie jednak poleca się ustawienie wspomnianych zmiennych środowiskowych.

Zasadniczo pilot jest obsługiwany z klawiatury, ale jeżeli konfiguracja systemu zapewnia wsparcie dla myszki, wówczas program można uruchomić z opcją -m i uaktywnić jej elementarną obsługę. Obsługę myszki można na bieżąco włączać lub wyłączać klawiszami Ctrl+ (Ctrl+|). Wygląd uruchomionego programu przedstawia rysunek 4.1.

W najwyższym wierszu ekranu wyświetlone są informacje o wersji programu i nazwa aktualnego katalogu. W dwóch najniższych znajduje się podpowiedź dotycząca obsługiwanych klawiszy.

W środkowej, głównej części ekranu jest wyświetlona lista nazw plików i katalogów. Katalogi oznaczone są napisem (dir), linki symboliczne mają zamiast rozmiaru napis --. Na tej liście nie są pokazywane pliki ukryte, czyli te, których nazwy rozpoczynają się od kropki. Aby je pokazać należy uruchomić program z opcją -a.
| Sesja Edycja Widok Zakładki | Ustawienia Pomoc | |
|-----------------------------|-------------------------------|--------------|
| UW PILOT 2.99 | BROWSER Dir: /etc | |
| | | |
| terminfo | (dir) texmf | (dir) |
| udev | (dir) ufw | (dir) |
| update-manager | (dir) update-notifier | (dir) |
| vim | (dir) w3m | (dir) |
| wpa_supplicant | (dir) X11 | (dir) |
| xdg | (dir) xml | (dir) |
| xulrunner-1.9 | (dir) adduser.conf | 3.0 KB |
| adjtime | 46 B aliases | 51 B |
| anacrontab | 395 B at.deny | 144 B |
| bash.bashrc | 1.7 KB bash_completion | 217 KB |
| bindresvport.blacklist | 332 B blkid.tab | 564 B |
| blkid.tab.old | 564 B bogofilter.cf | 6.8 KB |
| brlapi.key | 33 B brltty.conf | 16 KB |
| ca-certificates.conf | 4.9 KB crontab | 724 B |
| crypttab | 54 B debconf.conf | 3.0 KB |
| debian_version | 10 B deluser.conf | 600 B |
| e2fsck.conf | 34 B enscript.cfg | 5.1 KB |
| environment | 79 B fdmount.conf | 354 B |
| fpc.cfg П | 6.1 KB fstab | 659 B |
| ? Get Help 🛛 Quit | Launch - Prev Pg D Dele | ete 🕻 Copy 🗖 |
| 🖸 Goto 🛛 [View] | W Where is Spc Next Pg R Rena | ame 📱 Edit 🔽 |
| 🛃 📮 Powłoka | | ľ. |

Rysunek 4.1: Ekran startowy programu pilot

Katalog nadrzędny oznaczony jest dwoma kropkami znajdującymi się na początku listy.

Aby wejść do wskazywanego katalogu należy nacisnąć klawisz **Enter**. Program obsługuje klawisze kursora. Zastępczo można używać:

- spacja albo Ctrl+V następna strona,
- - (minus) albo Ctrl+Y poprzednia strona,
- Ctrl+F albo N następna kolumna,
- Ctrl+B albo P poprzednia kolumna,
- Ctrl+N następny wiersz,
- Ctrl+P poprzedni wiersz.

Pomoc dotycząca pozostałych obsługiwanych klawiszy jest wyświetlona w dwóch ostatnich wierszach. Aby uzyskać pokazaną na rysunku 4.1 funkcję *Goto* należy uruchomić program z opcją –j.

- ? albo Ctrl+G (Get Help) wyświetla ekran pomocy, po którym można poruszać się korzystając z klawiszy Ctrl+V (następna strona) i Ctrl+Y (poprzednia strona) i który można opuścić naciskając Ctrl+X,
- G(Goto) umożliwia szybką zmianę katalogu poprzez wpisanie nazwy,
- \mathbf{Q} (*Quit*) opuszcza program pilot,
- V (View) podgląd pliku (program honoruje zmienną środowiskową PAGER, jeśli jej nie znajdzie wywołuje pine),
- L (Launch) uruchamia zewnętrzne polecenie, naciśnięcie klawiszy Ctrl+X podczas edycji polecenia skopiuje nazwę wskazywanego

pliku do edytowanego wiersza, w ten sposób wydanie polecenia ln nazwa_pliku nazwa_pliku2 wymaga w gruncie rzeczy naciśnięcia ośmiu klawiszy,

- W (Where is) wyszukuje na liście plik o zadanej nazwie, wystarczy wpisanie dowolnego fragmentu nazwy, ignorowana jest różnica wielkości liter, opcja ta umożliwia również przejście na początek (Ctrl+Y) i koniec (Ctrl+V) listy.
- **D** (*Delete*) usuwa plik (nie ma możliwości usunięcia katalogu),
- \mathbf{R} (*Rename*) zmienia nazwę pliku,
- \mathbf{C} (*Copy*) kopiuje plik,
- $\mathbf{E} (Edit)$ wywołuje edytor (program honoruje zmienną środowiskową EDITOR, jeśli jej nie znajdzie wywołuje pico),

Niewątpliwą wadą programu jest to, że przeprowadzona w nim zmiana katalogu nie przekłada się na katalog roboczy systemu operacyjnego. W związku z tym polecenia wydawane po naciśnięciu klawisza L nie muszą dotyczyć katalogu widzianego w oknie programu. Aby zsynchronizować oba katalogi w pierwszej kolejności należy wydać polecenie cd. Jeżeli jednak do dyspozycji jest wyłącznie ten program to i tak może on znacznie ułatwić pracę.

4.4.3. Program mc - Midnight Commander

Legendarny program o kolosalnych możliwościach, a przy tym niebywale wygodny i łatwy w obsłudze. W 1986 John Socha w swoim programie, znanym pod komercyjną nazwą Norton Commander, zaproponował ideę dwóch paneli. Idea ta jest do dzisiaj spotykana w wielu programach do zarządzania plikami, w tym również w mc dla którego NC był niewątpliwie pierwowzorem.

W Midnight Commander zasadzie program może służyć jako punkt odniesienia. Osoba, która potrafi trybie W konsolowym przeprowadzić operacje dostępne w mc poprzez system menu może uważać siebie za eksperta. W tym miejscu można chyba zaryzykować stwierdzenie, że jeżeli jakaś funkcja nie została zaimplementowane w mc to najprawdopodobniej do niczego nie jest potrzebna.

Pomimo entuzjastycznego wstępu konieczne jest opisanie kilku problemów. Poprawne działanie programu mc wymaga zgodnych ustawień językowych środowiska i konsoli. We współczesnych, spolszczonych dystrybucjach systemu GNU/Linux po wydaniu polecenia:

\$ locale

zostanie najprawdopodobniej wyświetlona odpowiedź podobna do:

```
LANG=pl_PL.UTF-8
LC_CTYPE="pl_PL.UTF-8"
...
LC_IDENTIFICATION="pl_PL.UTF-8"
```

W starszych albo bardziej konserwatywnych systemach można spotkać:

```
$ locale
LANG=p1_PL.IS08859-2
LC_CTYPE="p1_PL.IS08859-2"
...
LC_MESSAGES="p1_PL.IS08859-2"
```

Zależnie od uzyskanego wyniku może okazać się konieczne ustawienie odpowiedniego kodowania w programie konsoli (terminala). Sposób osiągnięcia pożądanego efektu dla środowiska graficznego KDE3.x przedstawia rysunek 4.2, dla środowiska graficznego KDE4.x rysunek 4.3, a dla środowiska graficznego GNOME rysunek 4.4.

| Ustawienia Pomoc Ukryj pasek menu Karty Pasek przewijania Tryb pełnogkranowy Ctrl+Shift+F Brzęczyk Czcionka Czcionka | Tajskie (tis62 <u>0</u>) Tamilskie (tscii) Tureckie (p 1254) Tureckie (iso 8859- <u>9</u>) Unicode (wspólne) (iso-10646-ucs-2) Unicode (wspólne) (ucs2) Unicode (wspólne) (utf16) Unicode (wspólne) (utf7) |
|---|--|
| Kawiatura Zegtaw Bozmiar Zapisz jako domyślne Zapisz profil sesji Konfiguracja powiadomień Konfiguracja skrótów | V Unicode (wspólne) (utf8) Wietnamskie (cp 1258) Zachodnioeuropejskie (cp 1252) Zachodnioeuropejskie (ibm850) Zachodnioeuropejskie (iso 8859-1) Zachodnioeuropejskie (iso 8859-14) Zachodnioeuropejskie (iso 8859-15) Środkowoeuropejskie (cp 1250) Środkowoeuropejskie (ibm852) |
| 🔌 Konfiguracja: Konsola | 🗴 Środkowoeuropejskie (iso 8859- <u>2</u>) |

Rysunek 4.2: Ustawianie kodowania dla KDE3.x

Dla środowiska GNOME może okazać się konieczne wcześniejsze dodanie odpowiedniego kodowania.

W przypadku niezgodności ustawień wystąpią przekłamania polskich znaków, zaburzenia wyświetlania elementów semigraficznych i inne trudności rujnujące całą przyjemność korzystania z mc.

Jeżeli system ma zainstalowaną obsługę wielu języków to język polski można wybrać poleceniami typu:

```
export LANG=pl_PL.UTF-8
export KEYMAP=pl
```

| Plik Edycja Widok Przewijanie Zakładki Ustawienia Pomoc | |
|--|--|
| slawek@d620:~\$ | > |
| 🖉 <u>M</u> odyfikuj aktualny p | rofil 2 |
| 🖹 📃 Zapisz profil "Powłoka" - Konsola | Domyślne |
| Ogólne Karty Wygląd Przewijanie Wejście Zaawansowane | Zachodnioeuropejskie > |
| Funkcje konsoli | 6 Srodkowoeuropejskie 6 ISO 8859-2 Bałtyckie |
| ✓ Włącz migoczący <u>t</u> ekst | Południowoeuropejskie > cp 1250 |
| ✓ Włącz obsługę wyjścia klawiszami <u>C</u> trl+S i Ctrl+Q | <u>T</u> ureckie > |
| Włącz wyświ <u>e</u> tlanie dwukierunkowego tekstu | <u>Cyrylica</u> |
| | ○ Chińskie tradycyjne > |
| Obsługa myszy | Chińskie uproszczone > |
| Znaki tworzące słowa do zaznaczania podwójnym kliknięciem: | <u>K</u> oreańskie > |
| Kursor | _ Iapońskie > |
| Kurson | <u>G</u> reckie > |
| Migający kursor | <u>A</u> rabskie > |
| Kształt kursora: Blok 🗸 | → <u>Hebrajskie</u> → <u>6</u> UTF-8 |
| Dopasuj kolor kursora do aktualnego znaku | O Tajskie > O UTF-16 |
| Podany kolor kursora: | • <u>5</u> <u>U</u> nikod > () u <u>t</u> f7 |
| | Północne <u>S</u> aami > 🕐 u <u>c</u> s2 |
| Kodowanie | ○ Inne > ○ ISO 10646-UCS-2 |
| Domyślne kodowanie znaków: ISO-8859-2 4 W | Vybierz V |
| ✓ <u>O</u> K ✓ Zastosu | uj 🥝 Anuluj |

Rysunek 4.3: Ustawianie kodowania dla KDE4.x



Rysunek 4.4: Ustawianie kodowania dla GNOME

Drugie z nich jest odpowiedzialne za ustawienie polskiego mapowania klawiatury.

W przypadku niektórych modeli klawiatur generowane są nietypowe sekwencje znaków dla klawiszy sterujących i funkcyjnych. Program mc daje możliwość rozwiązania tego problemu "ucząc się" identyfikować te klawisze, tak jak to pokazano na rysunku 4.5 (aby aktywować menu należy nacisnąć klawisz **F9** albo kolejno klawisze **Esc**, **9**).

| Se | esja | Edycja | Widok | Zakładki | Ustawienia | Pomoc |
|-------|------------------------|--|--|---|--|---|
| | Lewy | y roc- | Plik | Poleo | enie | Opcje Prawy |
| | / /1 /11 /121 | Nazw | /a | Rozmiar ADRZEDN G G G | Cza 5 wrz 5 wrz 5 wrz | Konfiguracja ar CzasM Układ DN 00 12 maj 2008 Wyświetlanie znaków 00 21 mar 2008 Definiuj klawisze 00 21 mar 2008 |
| P s 1 | | []. kl [2. kl [3. kl [4. kl [5. kl [6. kl [7. kl [8. kl [10. k [11. k [12. k [12. k [12. k [13. k Naciś nie s albo | . funk . funk . funk . funk . funk . funk . funk 1. funh 1. funh 1. funh 1. funh 1. funh 1. funh | cyjny] cyjny] cyjny] cyjny] cyjny] cyjny] cyjny] cyjny] ccyjny] ccyjny] ccyjny] ccyjny] ccyjny] ccyjny] ccyjny] | [14. [15. [16. [17. [18. [20. [Klaw [Kurs [Kurs [Kurs [Kurs [Kurs [Kurs [Kurs]] [Kurs [Kurs]] | <pre>kl.funkcyjny] [Klawisz Home] kl.funkcyjny] [Klawisz Page Down] kl.funkcyjny] [Klawisz Page Up] kl.funkcyjny] [Klawisz Insert] kl.funkcyjny] [Klawisz Insert] kl.funkcyjny] [Klawisz Delete] kl.funkcyjny] [+ (numer.)] visz Backspace] [- (numer.)] visz End] [* (numer.)] visz End] [* (numer.)] visz Backspace] [- (numer.)] visz W Lewo] vor w dół] vor w ulewo] vor w prawo] vor w prawo] visz Następnie sprawdź, które Naciśnij spację na brakującym klawiszu viować. Poruszaj się klawiszem Tab.</pre> |

Rysunek 4.5: Definiowanie klawiszy w programie mc

Po pokonaniu tych trudności można rozpocząć poznawanie programu ${\tt mc}.$

Wygląd programu mc można dość swobodnie konfigurować we własnym zakresie, jednak najczęściej spotykany jest ten pokazany na rysunku 4.6.

Charakterystyczną belkę wskazującą jeden z plików (w przykładzie adjtime) można przerzucać pomiędzy panelami klawiszem **Tab**.

Panel w którym znajduje się belka jest panelem głównym. Katalog wybrany w panelu głównym jest równocześnie katalogiem bieżącym (roboczym) systemu operacyjnego. Drugi panel ma znaczenie pomocnicze. Na górnym obramowaniu paneli podawane są nazwy odpowiednich katalogów. Dla zilustrowanego przykładu panel główny wyświetla zawartość katalogu /etc, a panel pomocniczy katalogu /lib.

Naciskając klawisze $\mathbf{Ctrl} + \mathbf{U}$ można zamienić miejscami panel lewy i prawy.

| Sesja Edycja Widok | Zakładki U | Jstawienia | Pomoc | | | | | | |
|---|------------|------------|---------|------------------|---------|----|------|--------------------|----------|
| Lewy Plik | Polece | enie | Opcje | Prawy | | | | | |
| [<-/lib | 1 | | v>_ | < /etc | | | | <mark>v></mark> | n |
| Nazwa | Rozmiar | Cza | sM | Nazwa | Rozmiar | | Czas | sM | |
| /linux-r~modules | 4096 | 23 sie | 19:54 🔺 | /update-manager | 4096 | 25 | mar | 2009 | |
| /linux-s~nd-base | 4096 | 7 lut | 2009 🏼 | /update-notifier | 4096 | 7 | lut | 2009 | |
| /lsb | 4096 | 14 mar | 12:12 😐 | /vim | 4096 | 31 | sie | 11:45 | |
| /modules | 4096 | 23 sie | 19:53 | /w3m | 4096 | 23 | sie | 19:59 | |
| /security | 4096 | 23 sie | 19:55 | /wpa_supplicant | 4096 | 7 | lut | 2009 | |
| /terminfo | 4096 | 7 lut | 2009 | /xdg | 4096 | 22 | mar | 2009 | |
| /tls | 4096 | 7 lut | 2009 | /xml | 4096 | 23 | sie | 21:38 | |
| /udev | 4096 | 14 mar | 12:12 | /xulrunner-1.9 | 4096 | 23 | sie | 19:59 | |
| @cpp | 21 | 7 lut | 2009 🎆 | adduser.conf | 2975 | 7 | lut | 2009 | |
| *klibc-B~K09Y.so | 64612 | 20 sty | 2010 | adjtime | 46 | 5 | wrz | 23:23 | |
| *ld-2.7.so | 109152 | 21 maj | 20:12 | aliases | 51 | 7 | | 2009 | |
| @ld-linux.so.2 | 9 | 23 sie | 19:53 🎆 | anacrontab | | 5 | | 2007 | |
| libBrok~-2.7.so | 5440 | 21 maj | 20:12 🎆 | at.deny | 144 | 20 | lut | 2007 | |
| @libBrok~le.so.1 | 22 | 23 sie | 19:53 🎆 | bash.bashrc | 1733 | 12 | | 2008 | |
| libSegFault.so | 13696 | 21 maj | 20:12 🎆 | bash_completion | 216529 | 15 | | 2008 | |
| -> /etc/alternatives/cpp | | | | adjtime | | | | | |
| Porada: Polecenia powłoki działają tylko w lokalnym systemie plików. slawek@xordm:/etc\$ | | | | | | | | | |
| 🛃 💻 Powłoka | | | | | | | | | Ă |

Rysunek 4.6: Najczęściej spotykany wygląd programu mc

W najniższym wierszu ekranu jest opisane znaczenie klawiszy funkcyjnych **F1. F10**.

Program może być użytkowany również na klawiaturach bez klawiszy funkcyjnych (szczególnym przypadkiem mogą być klawiatury generujące nierozpoznawane kody dla klawiszy sterujących i funkcyjnych). Zamiast jednego z klawiszy funkcyjnych **F1**.. **F10** można w takim przypadku nacisnąć sekwencję klawiszy: jako pierwszy **Esc**, a później odpowiednio jeden z klawiszy **1**.. **9**, **0** z bloku alfanumerycznego.

Operacje przyporządkowane klawiszom funkcyjnym:

- F1 wyświetli obszerną, interaktywną pomoc dotyczącą programu, pomoc ma charakter kontekstowy zależny od obecnie wykonywanej operacji,
- F2 uaktywnia menu użytkownika. Szczegółowe omawiane tego zagadnienia wykracza poza ramy skryptu,
- F3 wywołuje podgląd wskazywanego pliku. Może do tego celu wykorzystać wewnętrzny, wbudowany program mcview o bardzo dobrych właściwościach. Parametry konfiguracyjne mogą zabraniać korzystania z mcview, wówczas program honoruje ustawienie zmiennej środowiskowej PAGER, a jeżeli jej nie znajdzie otwiera plik programem vi w trybie tylko do odczytu (program vi został omówiony w rozdziale poświęconym podstawowym edytorom plików tekstowych),
- F4 wywołuje edytor. Podobnie jak w przypadku F3 może

wywołać wewnętrzny, wbudowany bardzo dobry edytor mcedit, ale jeżeli konfiguracja na to nie zezwala wywoła edytor wskazywany przez zmienną środowiskową EDITOR. Jeżeli jej nie znajdzie wywoła vi (programy mcedit i vi zostały omówione w rozdziale poświęconym podstawowym edytorom plików tekstowych),

- F5, F6 Podział na panel główny i pomocniczy ma sens dla poleceń
 F5 Skopiuj i F6 Przenieś. Dla przykładu zilustrowanego rysunkiem 4.6 naciśnięcie klawisza F5 albo F6 spowoduje odpowiednio skopiowanie (cp) albo przeniesienie (mv) pliku adjtime z katalogu /etc do katalogu /lib,
- F7 zakłada nowy katalog (mkdir),
- F8 usuwa wskazywany plik albo katalog (rm),
- F9 wywołuje (uaktywnia) menu w najwyższym wierszu ekranu,
- F10 kończy pracę programu mc. Jeżeli program konsoli przechwytuje klawisz F10 można nacisnąć kolejno Esc, 0 (zero) albo po naciśnięciu F9 wybrać w menu głównym Plik |Zakończ.

Niektóre klawisze funkcyjne mogą zyskać inne znaczenie, jeżeli zostaną naciśnięte równocześnie z klawiszem **Shift**.

- Shift+F3 wywołuje alternatywny, surowy wariant podglądu pliku,
- Shift+F4 wywołuje edytor z nowym, pustym plikiem bez nazwy,
- Shift+F5 umożliwia wpisanie ścieżki i nazwy docelowej dla kopiowanego pliku,
- Shift+F6 podobnie jak Shift+F5 tylko dla przenoszonego pliku,
- Shift+F10 kończy pracę mc nie oczekując potwierdzenia.

Powyżej wiersza opisującego klawisze funkcyjne znajduje się wiersz poleceń. Można tam wpisać dowolne znane już polecenie, które zostanie wykonane po naciśnięciu klawisza **Enter**. Aby podejrzeć jego wyniki należy nacisnąć klawisze **Ctrl+O**, aby powrócić do paneli programu mc należy ponownie nacisnąć **Ctrl+O**. Podczas edycji wiersza poleceń wygodnie jest korzystać z kombinacji klawiszy **Alt+Enter**, po naciśnięciu której do wiersza poleceń kopiowana jest nazwa wskazywanego pliku.

Jeżeli wiersz poleceń jest pusty, to po naciśnięciu klawisza **Enter** reakcja programu jest zależna od wskazywanego pliku:

- jeżeli plik jest katalogiem, to program zmieni katalog,
- jeżeli plik jest programem (plikiem wykonywalnym binarnym albo skryptem), to program zostanie uruchomiony,
- w pozostałych wypadkach zostanie wywołane dla tego pliku polecenie skojarzone z jego rozszerzeniem. Skojarzenia można podejrzeć i przedefiniować korzystając z menu: F9 |Polecenie |Zmodyfikuj plik rozszerzeń.

Poniżej zebrano często używane, wygodne klawisze:

- Alt+Enter, Esc Enter, Ctrl+Enter (nie wszystkie kombinacje muszą działać) kopiuje nazwę wskazywanego pliku do wiersza poleceń (bardzo przydatna funkcja),
- Ctrl+O wywołuje podpowłokę, powrót do paneli mc nastąpi po ponownym naciśnięciu Ctrl+O,
- Ctrl+U zamienia panele miejscami,
- Ctrl+R odświeża zawartość panelu,
- Ctrl+I albo Tab przerzuca belkę pomiędzy panelami,
- Esc Tab autouzupełnienie. Ponieważ klawisz Tab jest wykorzystywany przez program mc do przerzucania belki pomiędzy panelami, aby uzyskać jego znaną już z konsoli funkcję autouzupełniania, należy posłużyć się sekwencją Esc Tab.
- Esc Esc zamyka wszelkie okna dialogowe, komunikaty o błędach, okna pomocy itp.
- Ctrl+S albo Esc S wywołuje okienko dialogowe poszukiwania pliku, należy po prostu wpisywać nazwę pliku, a wyszukiwanie będzie odbywało się na bieżąco,
- Alt+C albo Esc C wywołuje okienko dialogowe umożliwiające szybką zmianę katalogu poprzez wpisanie jego nazwy,
- Insert albo Ctrl+T ustawia albo zdejmuje znacznik dla wskazywanego pliku, później możliwe jest wykonanie operacji (na przykład kopiuj) dla zaznaczonych plików,
- + (plus na bloku numerycznym) albo **Alt**++ albo **Esc** + wywołuje okno dialogowe "Zaznacz" pliki można wpisać nazwę albo maskę nazw plików,
- - (minus na bloku numerycznym) albo **Alt+-** albo **Esc** – wywołuje okno dialogowe "Odznacz" o funkcji odwrotnej do +,
- * (gwiazdka na bloku numerycznym) albo Alt+* albo Esc * odwraca znaczniki plików.

Jest możliwe zaznaczanie lub odznaczanie katalogów. Należy w tym celu poprzedzić maskę znakiem / (ukośnik, slash).

Historia programu Midnight Commander jest dość długa, pamięta jeszcze bardzo powolne łącza internetowe, stąd też program unika generowania zbędnego ruchu w sieci. Jeżeli na skutek pracy w innym oknie terminala albo działania innych programów nastąpi zmiana zawartości katalogu, to nie zostanie to automatyczne zobrazowane w panelu. Można temu zaradzić użyciem klawiszy Ctrl+R co spowoduje odświeżenie treści wyświetlanej w panelu.

Dostępne w menu opcje **Lewy** i **Prawy** umożliwiają zmianę sposobu wyświetlania plików w odpowiednim panelu. Można wpłynąć na zawartość (wyświetlane informacje) oraz na klucz, wg którego pliki są uporządkowane

(posortowane). Jest możliwe również odfiltrowanie widocznych plików zgodnie z zadaną maską.

Szczegółowe omawianie wszystkich opcji dostępnych w menu wydaje się niecelowe. Znając już polecenia operujące na plikach, katalogach i dowiązaniach łatwo jest rozpoznać przeznaczenie poszczególnych opcji. Poniżej zostaną omówione tylko te ważniejsze i zarazem nie znajdujące prostej analogii z tematami poznanymi wcześniej, a których poznanie wymaga większego wysiłku niż kilka kliknięć myszką.

- **Polecenie** |**Porównaj katalogi** porównuje katalogi i raportuje różnice,
- **Polecenie** |**Historia poleceń** wyświetla historię poleceń wydanych z wiersza poleceń, możliwe jest wybranie i powtórzenie dowolnego z nich,
- Lewy (albo Prawy) |Połączenie FTP zestawia połączenie FTP z serwerem tego protokołu.

W wywołanym oknie dialogowym należy wpisać:

[użytkownik[:hasło]@]komputer[:port][zdalny katalog]

- jedynym wymaganym argumentem jest nazwa komputera albo jego adres IP,
- jeżeli zostanie pominięty użytkownik, wówczas zostanie użyty login z lokalnej maszyny,
- nie ma innej możliwości zadeklarowania użytkownika niż napis użytkownik@komputer,
- jeżeli zostanie pominięte hasło, a będzie ono konieczne to program o nie poprosi,
- można zestawić połączenie na porcie innym niż domyślny (jeżeli taka potrzeba wynika z konfiguracji serwera).

Lewy (albo Prawy) |Połączenie po powłoce – zestawia połączenie z innym komputerem z wykorzystaniem protokołu SSH albo RSH W wywołanym oknie dialogowym wpisujemy:

[użytkownik@]komputer[:opcje]

- jeżeli zostanie pominięty użytkownik, wówczas zostanie użyty login z lokalnej maszyny,
- nie ma innej możliwości zadeklarowania użytkownika niż napis użytkownik@komputer,
- program zapyta o hasło w drugim etapie,
- użycie opcji C włącza kompresję,
- użycie opcji **rsh** zestawia połączenie z wykorzystaniem protokołu **RSH** zamiast domyślnego **SSH**.

Niewątpliwą wadą oferowanego rozwiązania jest zestawianie połączenia wyłącznie na domyślnym porcie22– nie ma możliwości zmiany.

— Lewy (albo \mathbf{Prawy}) |Połączenie SMB – umożliwia połączenie

z komputerem pracującym pod kontrolą systemu operacyjnego Windows z włączoną usługą udostępniania plików i drukarek wykorzystując protokół SMB.

Wszystkie trzy połączenia umożliwiają jedynie zarządzanie plikami, nie dają możliwości wydawania poleceń i uruchamiania programów na zdalnej maszynie.

Rozdział 5

PRACA Z PLIKIEM TEKSTOWYM

| 5.1. | Pliki tekstowe | 0 |
|------|--|----|
| | 5.1.1. Wprowadzenie | 70 |
| | 5.1.2. Podstawowe operacje na plikach tekstowych 7 | 70 |
| 5.2. | Zaawansowana obsługa plików tekstowych 7 | '3 |
| | 5.2.1. Wprowadzenie | 73 |
| | 5.2.2. Programy do zaawansowanej obróbki plików | |
| | tekstowych | 73 |
| | 5.2.2.1. Programy diff i patch | 73 |
| | 5.2.2.2. Program grep i wyrażenia regularne | 74 |
| | 5.2.2.3. Program sed | 78 |
| 5.3. | Podstawowe edytory plików tekstowych | 6 |
| | 5.3.1. Wprowadzenie $\ldots \ldots \ldots$ | 36 |
| | 5.3.2. Edytor vi | 37 |
| | 5.3.3. Edytor nano |)3 |
| | 5.3.4. Edytor mcedit | 96 |
| | 5.3.5. Podsumowanie $\ldots \ldots \ldots$ |)4 |
| | | |

5.1. Pliki tekstowe

5.1.1. Wprowadzenie

Szczególną kategorię plików stanowią pliki tekstowe. Dane w nich zapisane mają naturalny i dobrze czytelny dla człowieka format. Wykorzystywane są w nich bajty z zakresu posiadającego czytelną reprezentację graficzną. Znaki te składane są w wiersze różnej długości zakończone znakiem końca wiersza (LF o kodzie ASCII 10) (autorzy w swej praktyce spotkali już pliki tekstowe składające się z jednego wiersza długości kilkudziesięciu kilobajtów).

Plikami tekstowymi są na przykład teksty źródłowe programów lub strony internetowe przygotowane w języku HTML. Nawet niniejszy skrypt został przygotowany jako zestaw plików tekstowych opisujących skład dokumentu dla systemu $T_{\rm E}X$.

Istnieje wiele narzędzi umożliwiających dostęp do plików tekstowych i ich obróbkę. Większość z nich należy do kategorii filtrów. Krótko zostaną omówione najważniejsze, rozpoczynając od prostszych.

5.1.2. Podstawowe operacje na plikach tekstowych

wc [opcje] [plik] (*word count*) – podaje elementarną charakterystykę plików tekstowych.

Przydatne opcje:

-c – zlicza liczbę znaków w pliku (podaje wielkość pliku),

-w – zlicza liczbę słów w pliku,

-1 – zlicza liczbę wierszy w pliku,

-L – znajduje najdłuższy wiersz w pliku i podaje jego długość.

Dopuszczalna jest dowolna kombinacja opcji. Pominięcie wszystkich opcji jest równoznaczne z uruchomieniem wc-lwc.

cat [plik] (*concat*) – uruchomiony bez żadnych argumentów dołącza standardowe wejście (klawiaturę) do standardowego wyjścia (monitora). W tym przypadku *dołącza* należy rozumieć *przepisuje*. Oczywiście możliwe jest przekierowanie pliku do standardowego wejścia.

Program uruchomiony z argumentem będącym nazwą pliku dołącza zawartość tego pliku do standardowego wyjścia (co należy rozumieć *wyświetla go na monitorze*).

Przydatne opcje:

- -n numeruje wiersze,
- -v wyświetla znaki niedrukowalne w czytelnej konwencji,

 -s – jeżeli w pliku występuje wiele pustych wierszy jeden pod drugim, to wyświetla tylko jeden.

W szczególnych przypadkach może zachodzić potrzeba podejrzenia zawartości plików binarnych w formacie ósemkowym albo szesnastkowym. Służą temu programy **od** (*octal dump*), **hexdump** i **xxd** (*hex dump*). Pierwsze dwa umożliwiają podejrzenie zawartości plików odpowiednio w formacie ósemkowym albo szesnastkowym. Trzeci program jest bardziej rozbudowany, potrafi przeprowadzać konwersję z formatu binarnego do szesnastkowego i odwrotnie.

Przeglądanie plików, które ze względu na swoją wielkość nie mieszczą się na jednym monitorze, możliwe jest na wiele sposobów. Możliwe jest wyświetlenie tylko fragmentów tych plików na przykład programami head albo tail, albo użycie programów stronicujących, z których dwa najbardziej znane to more i less (warto zwrócić uwagę na dość często spotykaną w GNU/Linuksie grę słów).

head [-wierszy] [plik] – wyświetla zadaną liczbę (domyślnie 10) pierwszych wierszy pliku.

tail [-wierszy] [plik] – wyświetla zadaną liczbę (domyślnie 10) ostatnich wierszy pliku.

Należy zauważyć, że połączenie powyższych dwóch programów pozwala wyświetlić dowolny fragment pliku. Przykład:

```
head -256 plik | tail 6
```

wyświetli wiersze od 251 do 256 z pliku plik.

more [plik] – starszy i prostszy program stronicujący, który wyświetla mieszczącą się na monitorze stronę tekstu. Po naciśnięciu **spacji** wyświetla następną stronę, po naciśnięciu klawisza **Enter** przewija tekst o jeden wiersz dalej. Tekst można cofać o cały ekran klawiszem **b**. Aby wyświetlić ekran pomocy programu należy nacisnąć klawisz **h**, a aby przerwać jego pracę – klawisz **q**.

Program potrafi przeszukiwać tekst. Aby rozpocząć przeszukiwanie należy nacisnąć klawisz / (ukośnik, *slash*) i wpisać poszukiwany tekst (albo wyrażenie regularne – wyrażenia regularne zostały omówione razem z programem **grep**). Przeszukiwanie rozpocznie się po naciśnięciu klawisza **Enter**. Kolejne wystąpienia poszukiwanego tekstu zostaną wyszukane po naciśnięciu klawisza **n**.

less [plik] – wbrew mylącej nazwie potrafi wyraźnie więcej niż more (co najważniejsze obsługuje klawisze kursora w sposób intuicyjny). Przejście na początek pliku umożliwia naciśnięcie klawisza <, a na koniec klawisza >. Obsługa klawiszy **spacja**, **Enter**, **h**, **q**, /, **n** odbywa się tak jak w more, jednak dodatkowo jest możliwe przeszukiwanie pliku wstecz. Aby rozpocząć poszukiwanie wstecz należy użyć klawisza ? zamiast /. Poszukiwanie poprzedniego wystąpienia następuje po naciśnięciu klawisza **N**. Oczywiście w obu przypadkach (/ i ?), klawisze **n** i **N** mogą być używane naprzemiennie.

Oba programy po naciśnięciu klawisza **v** wywołują domyślny edytor, którym jest najczęściej **vi**. Edytor **vi** został opisany obszerniej w rozdziale traktującym o podstawowych edytorach tekstu. Zakończenie pracy tego edytora bez zapisania przypadkowo wprowadzonych zmian w pliku, wiąże się z koniecznością użycia bardzo nietypowej sekwencji klawiszy:

Esc, : (dwukropek), q, ! (wykrzyknik), Enter

Oba programy (more i less) oczywiście mogą być składnikami potoku. Jeżeli w katalogu, w którym zapisano bardzo wiele plików zostanie użyte polecenie ls -l, wówczas efektem będzie jedynie szybkie przewinięcie ekranu. Wydając polecenie:

ls -l | less

można wygodnie przeglądać i przeszukiwać zawartość katalogu.

Należy mieć świadomość, że do wyświetlania stron manuala używany jest jeden z programów stronicujących (najczęściej **less**), stąd też znajomość podstawowej obsługi tychże jest zwyczajnie potrzebna.

5.2. Zaawansowana obsługa plików tekstowych

5.2.1. Wprowadzenie

Ze względu na bardzo istotne znaczenie plików tekstowych, system GNU/Linux oferuje bardzo bogaty zestaw programów przeznaczonych do ich obróbki. Oprócz tych omówionych w poprzednim rozdziale, należy wymienić sort, cut, nl, bc, dc, tr, uniq, jednak obszar ich zastosowania leży na marginesie tematyki objętej niniejszym skryptem i programy te nie będą tu omawiane. Zaleca się jednak zapoznanie z nimi.

Należy wspomnieć również o wszechstronnym programie AWK. Nazwa została utworzona z pierwszych liter nazwisk autorów: A. Aho, P. Weinberger i B. W. Kernighan. AWK jest nie tylko nazwą programu, ale również języka skryptowego ogólnego przeznaczenia interpretowanego przez ten program. Możliwości tego narzędzia są olbrzymie, jednak tak jak w przypadku programów wymienionych w poprzednim akapicie, pokrywają się w niewielkim stopniu z tematyką niniejszego skryptu i również nie będą omawiane. Obszerniejsze informacje można znaleźć w artykule "Awk - A Tutorial and Introduction". [23]

5.2.2. Programy do zaawansowanej obróbki plików tekstowych

5.2.2.1. Programy diff i patch

diff (*differences*) – znajduje różnice pomiędzy dwoma plikami. Narzędzie przeznaczone między innymi do porównywania tekstów źródłowych programów. Forma ogólna wywołania ma postać:

diff [opcje] plik_1 plik_2

plik_1, plik_2 oznaczają nazwy plików do porównania. Mogą być to również nazwy katalogów, a wówczas zostaną porównane pliki o zgodnych nazwach z obu katalogów.

Przydatne opcje:

- -a traktuj pliki jako tekstowe nawet jeżeli na takie nie wyglądają,
- -b ignoruj różnice wynikające z ilości znaków białych (spacji, tabulatorów),
- -B ignoruj różnice wynikające z ilości pustych linii,
- -i ignoruj różnice wynikające z wielkości znaków,
- -q raportuje sam fakt różnicy między plikami, bez wyświetlania szczegółów,
- -r porównuje katalogi rekursywnie,

-s – raportuje fakt identyczności plików – normalnie w takiej sytuacji program nie wyświetla żadnego komunikatu.

patch. Bardzo cenną zaletą programu diff jest to, że tworzony przez niego raport może być wykorzystany przez program **patch** do aktualizowania kodów źródłowych programów. Przykładowo jeden programista dysponuje plikami stare_zrodlo.c i nowe_zrodlo.c, podczas gdy u innego programisty znajduje się tylko stare_zrodlo.c. Pierszy z nich generuje raport_roznic.diff następującym poleceniem:

```
diff stare_zrodlo.c nowe_zrodlo.c >raport_roznic.diff
```

i wysyła go drugiemu programiście, który wykonuje:

```
patch stare_zrodlo.c raport_roznic.diff
mv -f stare_zrodlo.c nowe_zrodlo.c
```

albo:

```
patch stare_zrodlo.c raport_roznic.diff -o nowe_zrodlo.c
rm stare_zrodlo.c
```

Bez dodatkowych opcji (tak jak w pierwszym przykładzie) program patch modyfikuje stare_zrodlo.c. Opcja -o w drugim przykładzie powoduje zapisanie wyniku modyfikacji w pliku nowe_zrodlo.c. Więcej opcji programu patch można znaleźć w systemie pomocy.

W przypadku gdy same pliki źródłowe są bardzo duże, natomiast zmiany stosunkowo niewielkie, opisana operacja wiąże się z przesłaniem znacznie mniejszych ilości danych.

5.2.2.2. Program grep i wyrażenia regularne

grep (global / regular expression /print). Podstawowa forma wywołania:

```
grep [opcje] wzorzec [plik...]
```

nie sugeruje jak mocnym narzędziem jest **grep**. Cała tajemnica kryje się za dwoma literami nazwy **re** oznaczającymi *regular expressions*, czyli wyrażenia regularne, ale o tym za chwilę. Podstawowym zastosowaniem programu **grep** jest wyszukanie i wyświetlenie tych wierszy pliku tekstowego, które zawierają wzorzec. W najprostszym przypadku wzorcem jest po prostu konkretny napis. Przydatne opcje:

- -a traktuj plik binarny jako plik tekstowy,
- $-\mathbf{c}$ zlicza i wyświetla liczbę znalezionych wierszy (bez wyświetlania znalezionych wierszy),

- -e umożliwia zadanie wielu wzorców wg składni –e wzorzec –e wzorzec...
- -i ignoruj różnice wynikające z wielkości liter (utożsamiaj wielkie i małe litery),
- -m MAX po znalezieniu *MAX* liczby wierszy zawierających wzorzec, kończy pracę i nie przetwarza dalszej części pliku,
- -n dodatkowo wyświetla numery znalezionych wierszy,
- -q niczego nie wyświetla, fakt znalezienia wzorca można określić na podstawie statusu zakończenia programu, jeżeli znajdzie wzorzec kończy pracę bez przetwarzania dalszej części pliku,
- -v wyświetl (-c policz) wiersze nie zawierające wzorca,
- -w wzorzec oznacza całe słowo (nie może być fragmentem większego słowa),
- -x wzorzec oznacza cały wiersz (nie może być fragmentem dłuższego wiersza).

Dwie ostatnie opcje ułatwiają korzystanie z programu **grep** ale można je zrealizować budując odpowiednio wyrażenie regularne.

Wyrażenia regularne.

Termin *wyrażenie regularne* wywodzi się z teorii gramatyk i języków formalnych. Kryje się za nim łańcuch symboli opisujących wzorzec.

Wyrażenia regularne oprócz programu grep są wykorzystywane między innymi przez narzędzia sed i awk, a ich obsługa jest implementowana również w narzędziach interaktywnych (vi, more, less). Narzędzia te przeznaczone są do obsługi plików tekstowych. Podstawową jednostką, do której stosowane są wyrażenia regularne, jest jeden wiersz (linia) tekstu. Poszukiwane wyrażenie pasujące do wzorca musi więc znajdować się w obrębie jednego wiersza tekstu.

W wyrażeniach regularnych można wyróżnić 3 podstawowe składniki:

- zakotwiczenie,
- zbiór znaków, jaki może wystąpić na konkretnej pozycji,
- modyfikator określający powtarzanie poprzedzającego znaku (zbioru znaków).

Zakotwiczenie określa w jakim miejscu wiersza albo słowa musi znajdować się wyrażenie pasujące do wzorca. I tak:

- *wzorzec* wzorzec musi znajdować się na początku wiersza,
- \<wzorzec wzorzec musi znajdować się na początku słowa,
- wzorzec\> wzorzec musi znajdować się na końcu słowa,
- wzorzec\$ wzorzec musi znajdować się na końcu wiersza.

Tak więc ^\$ oznacza wzorzec, w którym koniec wiersza występuje zaraz za początkiem wiersza, czyli pusty wiersz, a polecenie:

grep -v ^\$ <plikWE >plikWY

przepisze plikWE do plikWY z pominięciem pustych wierszy.

Jeżeli poszukiwany jest konkretny napis, to należy się po prostu nim posłużyć. Przykład (trochę podchwytliwy):

grep nie <plikWE >plikWY

przepisze z plikWE do plikWY wiersze zawierające występujące po sobie trzy litery tworzące ciąg *nie*. No właś*nie*, *nie* słowo *nie*, tylko trzyliterowy ciąg *nie*, nawet jeżeli jest on częścią dłuższego słowa. Pierwszym pomysłem na wybra*nie* wierszy zawierających słowo *nie* jest dołącze*nie* spacji z przodu i z tyłu, oraz zamknięcie całości w apostrofy: '*nie* '. *Nie* jest to jednak pomysł najlepszy, po*nieważ nie* może wystąpić na początku wiersza – wtedy *nie* będzie spacji z przodu, albo może kończyć zda*nie* lub wiersz (no bo czemu *nie*?), wówczas *nie* będzie spacji za nim. Łatwo sobie wyobrazić znacz*nie* więcej pułapek interpunkcyjnych.

Poprawną obsługę takich sytuacji zapewniają kotwice początku i końca słowa:

- \<nie musi rozpoczynać słowo,
- nie\> musi kończyć słowo,
- $\langle nie \rangle > -$ musi być samodzielnym słowem.

Łatwiejszym sposobem użycia wzorca oznaczającego całe słowo jest posłużenie się opisaną wyżej opcją -w.

Kolejnym znakiem mającym szczególne znaczenie przy definiowaniu wzorców jest . (kropka). Oznacza ona dokładnie jeden, dowolny znak (oczywiście oprócz znaku zmiany wiersza). Tak więc wzorce $\hat{}.$ oraz $\langle <. \rangle >$ oznaczają odpowiednio wiersz składający się dokładnie z jednego znaku i jednoznakowe słowo.

Ponieważ kropka ma szczególne znaczenie, jeżeli poszukiwanym znakiem ma być właśnie kropka, trzeba użyć notacji \backslash . (*backslash* kropka). Podobnie w stosunku do innych znaków specjalnych, o których będzie za chwilę – w szczególności *backslash* \backslash trzeba notować jako podwójny *backslash* \backslash .

Jeżeli poszukiwany znak ma należeć do pewnego zbioru znaków, to można je wyspecyfikować w nawiasach kwadratowych. Na przykład [Nn]ie oznacza zarówno nie jak i Nie, natomiast \<[0123456789]\> oznacza jednocyfrową liczbę. Jeżeli specyfikowane znaki tworzą ciągły zakres, można użyć zapisu skrótowego korzystając ze znaku - (myślnik) (nazwa "myślnik" została użyta ze względu na kontekst, w rzeczywistości jest to znak minus: -, myślnik wygląda tak: -). Tak więc innym zapisem liczby jednocyfrowej będzie <[0-9]>, czyli znak - (myślnik) ma szczególne znaczenie. Aby znak ten był traktowany wprost jako myślnik (minus), powinien być poprzedzony przez *backslash* \ lub umieszczony jako pierwszy albo ostatni znak zestawu.

Ze względu na bardzo poprawną obsługę pułapek interpunkcyjnych, zapis $\langle [0-9] \rangle >$ oznacza liczbę jednocyfrową również ze znakiem plus lub minus, a w zasadzie poprzedzoną jakimkolwiek znakiem nie-alfanumerycznym.

Można formułować także bardziej zawiłe wzorce. Przykładowo [0-9A-Za-z] oznacza dowolną cyfrę lub literę (wielką albo małą).

W tym momencie można pokusić się o skonstruowanie wzorca, do którego pasują daty zapisane zgodnie z polską normą:

[0-9] [0-9] [0-9] \. [01] [0-9] \. [0-3] [0-9]

Wzorzec nie jest *szczelny*, bowiem zaakceptuje również jako datę 9999.19.39. Wyrażenia regularne nie umożliwiają kontroli zakresów.

Teraz kolej na drugie znaczenie symbolu [^]. Znak ten umieszczony jako pierwszy znak zestawu (na przykład [[^]0-9] oznacza negację zestawu (w przykładzie dowolny znak poza jakąkolwiek cyfrą).

Jeżeli znak ^ nie znajduje się na pierwszej pozycji zestawu, traktowany jest literalnie i grep nie przypisuje mu żadnego szczególnego znaczenia. Opisywany jako następny program sed używa dodatkowo zestawienia ^LITERA. Znaczenie takiego zapisu zostanie wytłumaczone we właściwym czasie.

Kolejnym bardzo ważnym znakiem używanym przy definiowaniu wyrażeń regularnych jest * (gwiazdka). Oznacza ona, że poprzedzający ją znak może wystąpić dowolnie wiele razy – w tym zero razy (może nie wystąpić w ogóle). Na przykład zapis $\langle [A-Za-z_] [0-9A-Za-z_]* \rangle$ oznacza, że pierwszym znakiem może być dowolna duża litera, mała litera albo znak podkreślenia ($[A-Za-z_]$), po czym występuje dowolnie dużo cyfr, liter i znaków podkreślenia ($[0-9A-Za-z_]*$). Całość musi stanowić samodzielne słowo ($\langle \langle \rangle$). Do wzorca tego pasują więc identyfikatory zmiennych używane w językach programowania.

Podobne znaczenie do * (gwiazdki) ma znak + (plus). Oznacza on co najmniej jedno wystąpienie poprzedzającego znaku (zestawu).

Warto wspomnieć o jeszcze jednej zdolności wyrażeń regularnych. W odróżnieniu od * (gwiazdki) i + (plusa), które nie ograniczają liczby powtórzeń znaków, możliwe jest narzucenie dopuszczalnego zakresu powtórzeń. Służy do tego notacja $\{i\}$.

Przykłady: — [0-9] \{1, \} – musi wystąpić przynajmniej jedna \{1, \} cyfra [0-9],

- $[a-z] \{2,4\}$ dowolna mała litera [a-z] musi wystąpić 2, 3 albo 4 razy $\{2,4\}$,
- $[A-Z] \setminus \{3\}$ dokładnie trzyliterowy $\setminus \{3\setminus\}$ ciąg dużych liter [A-Z].

Każdy kto chce dowiedzieć się więcej o wyrażeniach regularnych może skorzystać z artykułu "Regular Expressions"[24].

5.2.2.3. Program sed

Drugim narzędziem o bardzo dużych możliwościach obróbki plików tekstowych jest **sed** (*stream editor*) – nieinteraktywny edytor strumienia danych. Podobnie jak w przypadku programu **grep** sposób wywołania nie sugeruje jego potęgi:

```
sed [opcje] polecenie [plik...]
```

Bardziej przydatne opcje:

- -e umożliwia zadanie wielu poleceń w jednym wywołaniu,
- -n nie będzie niczego wypisywał na standardowym wyjściu, ma to sens w połączeniu z p (szczegóły dalej).

Najbardziej istotnym poleceniem wykonywanym przez sed jest s, czyli substitute (zamień, zastąp).

Przykład:

echo "Ala ma Asa" | sed s/A/O/

wyprowadzi na monitor napis:

Ola ma Asa

podczas gdy wynikiem:

echo "Ala ma Asa" | sed s/A/O/g

będzie

Ola ma Osa

Różnica jest oczywista. Flaga g(global) powoduje, że przetwarzany jest cały łańcuch wejściowy, jej brak powoduje zamianę jedynie pierwszego wystąpienia poszukiwanego fragmentu.

sed nie jest rekursywny, to znaczy że jeżeli w wyniku zamiany powstanie łańcuch, który mógłby być w dalszym ciągu przetwarzany według zadanego schematu, to dalsze zmiany już nie będą wykonywane. Tak więc:

echo 12333 | sed s/23/12/

zakończy się wynikiem 11233, a nie jak można by przypuszczać 11112, natomiast

echo "Ala ma Asa" | sed s/A/A/g

nie spowoduje zawieszenia komputera.

Zwyczajowo przyjęło się stosowanie znaku / (*slash*, ukośnik) jako separatora w poleceniu substitute. W rzeczywistości **sed** traktuje jako separator pierwszy znak występujący po literze **s** i może to być dowolny znak (musi oczywiście wystąpić łącznie trzy razy). Może to istotnie ułatwić konstrukcję polecenia jeżeli w poszukiwanym wzorcu lub w łańcuchu zastępującym występuje znak /. Wystarczy porównać dwa poniższe, równoważne wywołania:

```
sed 's/\/usr\/local\/bin/\/common\/bin/' plikWE >plikWY
sed 's_/usr/local/bin_/common/bin_' plikWE >plikWY
```

W pierwszym przypadku, aby wymusić traktowanie znaku / jako znaku występującego w ścieżce, trzeba było stosować notację \/, co umożliwiło programowi **sed** odróżnienie go od znaku / pełniącego funkcję separatora. W drugiej wersji funkcję separatora zlecono znakowi _ (podkreślenie), więc konstrukcja całego wyrażenia stała się znacznie prostsza i czytelniejsza.

sed potrafi interpretować wyrażenia regularne. Dodatkowo obsługiwany jest znak & o specjalnym znaczeniu. Rozważmy następujący przykład:

```
sed 's/ [a-z][a-z]*/ (& )/' plikWE >plikWY
```

Czytane będą dane z pliku plikWE wiersz po wierszu. W każdym wierszu wyszukiwany będzie pierwszy łańcuch zaczynający się spacją, po czym ma wystąpić mała litera [a-z], po której nastąpi 0 (zero) lub więcej małych liter [a-z]*, jeżeli taki łańcuch zostanie znaleziony to & przyjmie jego wartość i w rezultacie na wyjście zostanie wypisane: spacja, nawias otwierający, znaleziony łańcuch, spacja, nawias zamykający. Przykład:

```
echo "Ala ma Asa i kota" | sed 's/ [a-z][a-z]*/ (& )/'
działa tak: Ala ( ma ) Asa i kota, podczas gdy:
echo "Ala ma Asa i kota" | sed 's/ [a-z][a-z]*/ (& )/g'
działa tak: Ala ( ma ) Asa ( i ) ( kota ).
```

W dotychczasowych przykładach nie występowały znaki o znaczeniu specjalnym dla Uniksa, więc polecenie mogło być przekazane bez cytowania. W tym przykładzie wystąpiły takie znaki (spacja, gwiazdka, ampersand, nawias otwierający i zamykający). Aby zapobiec interpretacji wyrażenia przez powłokę, tym razem konieczne było cytowanie (zamknięcie wyrażenia

w apostrofy). Lepiej przyjąć cytowanie jako regułę, uniknie się wówczas wielu niespodzianek.

Symbol & może wystąpić wielokrotnie w wyrażeniu zastępującym. Przykład:

```
echo "1 12 123 abc" | sed 's/[0-9][0-9][0-9]*/& &/g'
```

spowoduje 1 12 12 123 123 abc (każdy fragment składający się co najmniej z dwóch cyfr zostanie wyprowadzony dwa razy (ze spacją w środku)).

Domyślnie **sed** wyprowadza na wyjście wszystko co przeczyta, (albo w formie oryginalnej, albo zmodyfikowanej). Wywołany z opcją **-n** będzie zachowywał się inaczej. Przykładowo można programem **sed** uzyskać taki sam efekt jak w przypadku programu **grep**:

grep wzorzec <plikWE sed -n 's/wzorzec/&/p' plikWE</pre>

Działanie programu grep jest już znane: czyta plikWE wiersz po wierszu i wyprowadza na wyjście te wiersze, w których znajduje się wzorzec. sed wywołany w sposób pokazany tutaj, daje taki sam efekt: plikWE jest czytany wiersz po wierszu. Jeżeli w wierszu znajduje się wzorzec, to jest on zastępowany przez & (czyli przez samego siebie) i wówczas polecenie p wyprowadza tak "zmodyfikowany" wiersz wejściowy na wyjście. Jeżeli w powyższym przykładzie nie wystąpiłaby opcja -n, to na wyjściu pojawiłby się cały plikWE, a wiersze zawierające wzorzec zostałyby zdublowane. Natomiast:

```
sed 's/dowolnyWzorzec/&/' plikWE
```

da efekt dokładnie taki sam jak cat, czyli wyprowadzona zostanie cała zawartość plikWE (brak opcji -n powoduje wyprowadzenie wszystkiego co sed przeczyta, a to czy substitute dokona niezauważalnej zamiany dowolnegoWzorca na niego samego, nie ma w tym przypadku żadnego znaczenia). Można to zrobić prościej, ale o tym przy poleceniu print.

Załóżmy, że zaszła potrzeba wymiany w źródle programu napisanego w języku Pascal słów BEGIN i END na słowa pisane małymi literami. Najprostsze rozwiązanie to przepuścić plik przez seda dwukrotnie:

```
sed 's/BEGIN/begin/' <stary.pas | sed 's/END/end/' >nowy.pas
```

jednak **sed** oferuje bardziej wydajną możliwość wymiany wielu wzorców w jednym przebiegu:

```
sed -e 's/BEGIN/begin/' -e 's/END/end/' <stary.pas >nowy.pas
```

Jeżeli zamieniany jest tylko jeden wzorzec, to dopuszczalne jest pominięcie opcji –e (tak jak to miało miejsce we wcześniejszych przykładach). Podobne zachowanie wykazywał poznany wcześniej programu grep.

Drugą cechą, czyniącą z programu **sed** bardzo mocne narzędzie, jest możliwość operowania na wybranym fragmencie pliku. Najistotniejsze możliwości to:

- wybranie jednego wiersza przez podanie jego numeru,
- wybranie fragmentu pliku (numery pierwszego i ostatniego przetwarzanego wiersza),
- przetwarzanie wyłącznie wierszy zawierających wzorzec,
- przetwarzanie od początku pliku do napotkania wyrażenia regularnego,
- przetwarzanie od wyrażenia regularnego do końca pliku,
- przetwarzanie fragmentu określonego przez początkowe i końcowe wyrażenie regularne.

Przykłady:

sed '3 s/[0-9][0-9]*//' plikWE >plikWY

z trzeciego wiersza plikWE zostanie usunięta pierwsza przynajmniej jednocyfrowa liczba (mówiąc precyzyjnie zostanie zastąpiona przez pusty łańcuch),

sed '/^#/ s/[0-9][0-9]*//' plikWE >plikWY

wyszuka w plikWE wszystkie wiersze zaczynające się od # (dokładnie to oznacza wyrażenie /^#/) - dalej jak w poprzednim przykładzie,

sed '1,100 s/A/a/' plikWE >plikWY

w każdym z pierwszych stu wierszy plikWE pierwsze wystąpienie wielkiej litery A zostanie zamienione na małą literę a,

```
sed '101,$ s/A/a/' plikWE >plikWY
```

począwszy od wiersza 101 do końca pliku... dalej jak w poprzednim przykładzie,

```
sed '/start/,/stop/ s/#.*//' plikWE >plikWY
```

począwszy od wiersza zawierającego słowo **start** do wiersza zawierającego słowo **stop** znajdź znak **#** i zamień wszystko do końca wiersza na łańcuch pusty (znaczy zatrzyj komentarze w sensie **bash**),

sed '1,/stop/ s/#.*//' plikWE >plikWY

różne sposoby określania zakresu mogą być mieszane. (od wiersza pierwszego do wiersza zawierającego słowo stop),

sed -e '1,/start/ s/#.*//' -e '/stop/,\$ s/#.*//' plikWE >plikWY

zacznij od pierwszego wiersza i usuwaj komentarze **bash** aż do wiersza ze słowem **start**, potem do napotkania słowa **stop** nic nie zmieniaj, a potem do końca pliku też usuwaj komentarze,

Wiersze określone jako początek lub koniec zakresu (niezależnie od sposobu określenia) są uwzględniane w przetwarzaniu.

Polecenie substitute jest niewątpliwie najważniejsze w repertuarze programu sed, ale nie jedyne. Teraz na przykładach kilka słów o delete:

sed '11,\$ d' plikWE

pomiń przy wyprowadzaniu na wyjście wiersze od 11 do końca pliku, efekt równoważny z head <plikWE albo head -10 <plikWE.

```
sed '1,/^$/ d' plikWE
```

wyrażenie /^\$/ należy rozumieć szukaj / na początku wiersza ^ końca wiersza \$ (czyli szukaj pustych wierszy). Całość oznacza więc pomiń wiersze od pierwszego do pierwszego pustego (jeżeli pierwszym pustym wierszem jest pierwszy wiersz pliku to nie wyprowadzi nic),

```
sed '/^#/ d' plikWE
```

pomiń wszystkie wiersze rozpoczynające się #.

Ostatnie polecenie (sed '/^#/ d') może być zaczątkiem polecenia usuwającego komentarze ze skryptu. Jeżeli znak # jest pierwszym znakiem w wierszu, to taki wiersz zostanie pominięty podczas wyprowadzania na wyjście.

Ambitniejszym zadaniem będzie usunięcie komentarzy nie znajdujących się na początku wiersza tylko gdzieś dalej? Zostanie to zademonstrowane w kilku małych krokach:

sed -e 's/#.*//' -e '/^\$/ d'

Działa podobnie, ale robi już trochę więcej. Jeżeli -e 's/#.*//' znajdzie / łańcuch rozpoczynający się do od #, po czym następuje 0 (zero) lub dowolnie wiele dowolnych znaków .*, to zamieni go na łańcuch pusty. Następne polecenie było już omawiane: -e '/^\$/ d' pomija wiersze puste.

Jakie są podobieństwa? Jeżeli wiersz rozpoczyna się od znaku #, to tak jak poprzednio, zostanie całkowicie pominięty.

Jakie są różnice? Zostaną wycięte komentarze, które nie znajdowały się na początku wiersza, bez zmiany treści przed komentarzem oraz zostaną wycięte wiersze puste z pliku wejściowego.

Trzecia, jeszcze bardziej rozbudowana forma:

sed -e 's/#.*//' -e 's/[^I]*\$//' -e '/^\$/ d'

Pierwszy (-e 's/#.*//') i ostatni (-e '/^\$/ d') fragment jest już znany.

Co robi -e 's/[^I]*\$//'? Jeżeli bezpośrednio przed końcem wiersza \$ występuje dowolnie wiele (*) znaków zawartych w nawiasach kwadratowych [^I], to zastąp taki fragment łańcuchem pustym.

Co znajduje się w nawiasach kwadratowych? Pierwszym znakiem jest spacja, potem następują dwa znaki ^ (daszek, *caret*) i duża litera I. Takie połączenie rozumiane jest przez program **sed** (zresztą nie tylko przez niego) jako znak tabulacji.

Jaki jest efekt końcowy? W pierwszej kolejności wszystko co zaczyna się od # jest do końca wiersza usuwane. Jeżeli # był poprzedzony ciągiem spacji lub tabulacji (albo po prostu były one na końcu wiersza), to zostaną one usunięte w drugim kroku. Jeżeli w rezultacie powstał wiersz pusty, to nie pojawi się on na wyjściu.

Ze sztuczki sed -e 's/[^I]*\$//' można korzystać przy innych okazjach. Na przykład mcedit (edytor Midnight Commandera) zostawia spacje lub tabulacje w wierszach nie zawierających niczego innego, lub zostawia je na końcu wiersza – w zasadzie bez żadnej potrzeby. Przepuszczając źródło programu przez taki filtr można go odchudzić.

Zanim zostanie omówione następne polecenie (print), należy przypomnieć, że polecenie:

sed '11,\$ d' plikWE

daje taki sam efekt jak jedno z poleceń:

```
head <plikWE
head -n 10 <plikWE
```

czyli za pomocą programu sed można osiągnąć wyniki działania programu head. A jak osiągnąć rezultaty programu tail? Program sed samodzielnie nie daje takiej możliwości, ale jest to możliwe przy pomocy innych znanych programów.

Rozważmy następującą konstrukcję:

sed "1,\$(('wc -l plikWE' - 10)) d" plikWE

polecenie wc z opcją –1 zlicza wiersze występujące w plikWE. Ponieważ całość 'wc –1 plikWE' zamknięta jest w odwrotne apostrofy, powłoka przed wywołaniem programu sed zastąpi ten fragment wynikiem jego działania. Jeżeli plikWE miał 64 wiersze to można wyobrazić sobie, że przez chwilę zostanie uzyskany następujący zapis:

sed "1,\$((64 - 10)) d" plikWE

W drugim kroku powłoka wyliczy wartość wyrażenia \$((64 - 10)), czyli uzyskujemy:

sed "1,54 d" plikWE

i dalej wszystko jest już jasne. Pominięte zostaną wiersze od 1 do 54 i zostanie wyprowadzonych 10 ostatnich wierszy.

Rozwiązanie ma kilka słabych punktów. Po pierwsze jest mało efektywne – plikWE czytany jest 2 razy. Po drugie nie działa dla plików, które mają mniej niż 10 wierszy. Ale prostą namiastkę programu tail udało się osiągnąć.

Wspomniany print był już raz użyty w połączeniu z substitute po wyciszeniu wyjścia opcją -n. print może jednak funkcjonować jako samodzielna komenda. Sygnalizowana, prostsza realizacja programu cat za pomocą programu sed może wyglądać tak:

```
sed -n p plikWE
```

podczas gdy:

```
sed p plikWE
```

spowoduje dwukrotne wyprowadzenie każdego wiersza.

print oczywiście może być łączony z zakresami (w dowolnym znaczeniu), czyli:

sed -n '1,10 p' plikWE

jest kolejną realizacją programu head, natomiast:

sed -n '/wzorzec/p' działa dokładnie tak samo jak grep 'wzorzec'.

Powstaje pytanie: jak uzyskać efekt działania programu grep z opcją-v? Można to zrobić tak:

```
sed -n '/wzorzec/ !p' plikWE
```

co należy rozumieć *nie drukuj* !p wierszy zawierających wzorzec (ale drukuj wszystkie pozostałe).

sed ma jeszcze jedną wartą odnotowania umiejętność: możliwość zamiany znak po znaku. Przykłady:

```
sed 'y/aeiou/AEIOU/' plikWE
```

spowoduje zamianę wszystkich małych samogłosek na duże, natomiast

sed 'y/ąćęłńóśżźĄĆĘŁŃÓŚŹŻ/acelnoszzACELNOSZZ/' plikWE

zamieni polskie znaki na ich łacińskie pierwowzory.

Przez analogię można skonstruować polecenie konwertujące polskie znaki ze standardu windows-1250 na iso-8859-2.

Polecenie y wymaga, aby łańcuchy znaków zastępowanych i zastępujących były tej samej długości.

Na zakończenie pozostaje szczątkowa wzmianka o poleceniach append, insert i change. Te trzy polecenia można stosować dokładnie tak samo jak inne opisywane dotychczas. Przykłady:

```
sed '10 i Nastepny wiersz to wiersz dziesiaty' plikWE
```

przepisuje plikWE na wyjście do wiersza 10, jako dziesiąty pojawi się na wyjściu wiersz o treści Nastepny wiersz to wiersz dziesiaty, za nim jako wiersz jedenasty pojawi się wiersz, który w plikWE był dziesiąty i dalej plikWE zostanie do końca wysłany na wyjście. Przez analogię:

```
sed '/Ala/ i W nastepnym wierszu bedzie Ala' plikWE
```

wszystkie wiersze zawierające słowo Ala zostaną poprzedzone wierszem W następny wierszu bedzie Ala.

Następne przykłady chyba nie wymagają dodatkowego komentarza:

sed '10 a Poprzedni wiersz to wiersz dziesiaty' plikWE

sed '/Ala/ a W poprzednim wierszu byla Ala' plikWE

sed '10 c Wiersz dziesiaty zostal wymieniony' plikWE

sed '/Ala/ c Zgadnijcie kto byl w tym wierszu?' plikWE

Pojawiają się różnice w działaniu poleceń append i insert, a poleceniem change jeżeli operujemy na zakresie pliku:

sed '10,15 i pojawie sie przed kazdym wierszem zakresu' plikWE

sed '10,15 a pojawie sie po kazdym wierszu zakresu' plikWE

sed '10,15 c caly zakres zastapi ten jeden wiersz' plikWE

I działa to dokładnie tak samo, jeżeli zakres zostanie zdefiniowany przez wyrażenia wyszukujące.

Więcej na temat programu **sed** można dowiedzieć się z artykułu "Sed -An Introduction and Tutorial" [25].

5.3. Podstawowe edytory plików tekstowych

5.3.1. Wprowadzenie

Oprócz programów omówionych w poprzednich rozdziałach podstawowym narzędziem codziennej pracy programisty jest edytor.

W systemie GNU/Linux dostępnych jest bardzo wiele edytorów i to zarówno przeznaczonych do pracy w trybie graficznym (kate (dla środowiska KDE), gedit (dla środowiska GNOME), gvim), jak i tekstowym.

Obsługa edytorów przeznaczonych do pracy w trybie graficznym opiera się obecnie na podobnych wzorcach i oferuje zbliżone funkcje, których poznanie nie nastręcza trudności, ponieważ są one dostępne poprzez systemy menu i menu kontekstowego (powiązanego zazwyczaj z prawym klawiszem myszy). Każdy kto posługiwał się chociaż jednym z nich, w krótkim czasie przystosuje się pracy z każdym innym opartym na takiej samej koncepcji. Z tego powodu ta kategoria edytorów nie zostanie omówiona.

Wbrew pozorom edytory pracujące w trybie tekstowym są i pozostaną narzędziem niezbędnym programiście z kilku powodów. Oto kilka przekonujących:

- w przypadku instalacji serwerowych często pomija się instalację trybu graficznego,
- w sytuacjach awaryjnych może okazać się, że nie można uruchomić trybu graficznego i konieczne jest przeprowadzenie konfiguracji systemu w trybie tekstowym,
- w przypadku zdalnego dostępu do innego komputera, tryb tekstowy stawia znacznie mniejsze wymagania (na przykład co do przepustowości łącza) i pozostaje nadal podstawowym trybem pracy.

Od typowego edytora programisty oczekuje się kilku cech wspomagających tworzenie czytelnego, przejrzystego kodu źródłowego. Do najważniejszych należą:

- możliwość zmiany szerokości znaku tabulacji,
- możliwość zmiany wielkości wcięcia bloku tekstu,
- możliwość wyszukiwania par nawiasów,
- operacje na bloku tekstu,

— automatyczne utrzymywanie wcięcia z poprzedniego wiersza.

Te cechy będą przedmiotem zainteresowania podczas omawiania wybranych edytorów.

W systemie GNU/Linux użytkownik może zadeklarować swój ulubiony edytor w zmiennej środowiskowej EDITOR. Można to osiągnąć wydając polecenie:

export EDITOR=mcedit

które spowoduje, że domyślnie będzie uruchamiany edytor mcedit. Najwygodniej takie polecenie zapisać na stałe w pliku ~/.bashrc (albo innym pełniącym podobną funkcję np. ~/.profile). Będzie ono wówczas automatycznie wykonywane przy każdym logowaniu się do systemu.

Jednym z najbardziej rozbudowanych i wszechstronnych edytorów dostępnych dla systemu GNU/Linux jest emacs – poświęcono mu osobny rozdział. Tutaj zostaną omówione trzy wybrane, prostsze edytory. Będą to kolejno vi, nano, mcedit. Niezbędnych jest kilka słów wprowadzenia dla każdego z nich, ponieważ edytory te nie obsługują myszki albo obsługują ją w bardzo ograniczonym zakresie.

5.3.2. Edytor vi

vi to ciągle funkcjonująca nazwa starszej wersji edytora. Obecnie kryje się za nią ulepszony program Vim (*vi improved*). Nazwa vi została wyprowadzona od słowa *visual*. Jest to jeden z najbardziej zasłużonych dla rozwoju Uniksa edytorów. Spotykany jest w każdej dystrybucji systemu GNU/Linux. Stąd też opanowanie jego obsługi chociażby w podstawowym zakresie jest niezbędne.

Edytor ten często instalowany jest w wersji minimalnej – nawet bez systemu pomocy i jak najbardziej celowe jest jego doinstalowanie. W dystrybucjach obsługujących *debianowy* system pakietów (w tym również w rodzinie *Ubuntu*) służy temu polecenie:

sudo aptitude install vim-runtime

Edytor można uruchomić poleceniami:

```
vi
vi nazwa_pliku
vi +numer_wiersza nazwa_pliku
```

W ostatnim przypadku kursor zostanie ustawiony w żądanym wierszu. Sam + (plus) bez argumentu numer_wiersza ustawia kursor na końcu pliku.

Wygląd programu po uruchomieniu bez argumentów pokazuje rysunek 5.1.

Tak jak już było to napisane przy omawianiu programów stronicujących, naciśnięcie klawisza v podczas przeglądania pliku w programach more i less uruchamia edytor domyślny, którym najczęściej jest właśnie vi (chyba że zadeklarowano inny w zmiennej środowiskowej EDITOR).

Jeżeli system pomocy jest już zainstalowany można z niego skorzystać wpisując :help (dwukropek help) i naciskając klawisz Enter (rys. 5.2).



Rysunek 5.1: Ekran startowy edytora Vim

| help.txt* For | Vim version 7.1. Last change: 2006 Nov 07 | 4 |
|---|---|-------|
| | VIM - main help file | |
| Move around: Close this window: Get out of Vim: | Use the cursor keys, or "h" to go left, h l "j" to go down, "k" to go up, "l" to go right. j Use ":q <enter>". Use ":qa!<enter>" (careful, all changes are lost!).</enter></enter> | |
| Jump to a subject: With the mouse: Jump back: | Position the cursor on a tag (e.g. bars) and hit CTRL-]. ":set mouse=a" to enable the mouse (in xterm or GUI). Double-click the left mouse button on a tag, e.g. bars . Type CTRL-T or CTRL-0 (repeat to go further back). | 11111 |
| Get specific help: | It is possible to go directly to whatever you want help on, by giving an argument to the :help command. It is possible to further specify the context: *help-context* WHAT PREPEND EXAMPLE Normal mode command (nothing) :help x | |
| help.txt [Help][R0] | 1,1 Тор | |
| [No Name] | 0,0-1 All | |

Rysunek 5.2: Pierwszy ekran pomocy edytora vi

Najpóźniej razem z systemem pomocy zostanie zainstalowana dokumentacja. Przykładowo może ona znajdować się w katalogu /usr/share/vim/vimVV/tutor, gdzie VV oznacza numer wersji. Dokumentacja ma postać zwykłych plików tekstowych, które można otwierać i czytać wykorzystując program vi. Szczególnie godny polecenia jest plik tutor.pl (albo tutor.pl.utf-8), który umożliwia dość dobre poznanie edytora vi mniej więcej w ciągu 30 minut.

Koncepcja programu vi jest starsza niż układ obecnie używanych klawiatur, stąd też jego obsługa przy pierwszym kontakcie bywa szokująca.

Edytor może znajdować się w jednym z dwóch trybów: trybie edycji albo trybie poleceń. Bezpośrednio po uruchomieniu program znajduje się zawsze w trybie poleceń. Jest wiele sposobów przejścia z trybu poleceń do trybu edycji – zostaną omówione dalej. Aby powrócić z trybu edycji do trybu poleceń należy nacisnąć klawisz **Esc**. Klawisz **Esc** w trybie poleceń nie ma żadnego znaczenia, ale jego naciśnięcie jest sygnalizowane dźwiękowo.

Ponieważ jest to permanentny problem ponownie należy przypomnieć, że aby opuścić edytor vi bez zapisywania zmian w tekście, należy kolejno nacisnąć klawisze:

Esc, : (dwukropek), q, ! (wykrzyknik), Enter.

(vi w poleceniach rozróżnia wielkie i małe litery – należy zwrócić na to uwagę).

Obecne wersje edytora (Vim) obsługują klawisze kursora, ale historycznie do tego celu służyły klawisze **hjkl**. Aby łatwo zapamiętać ich funkcje wystarczy przyjąć, że klawisz **j** przypomina swoim wyglądem strzałkę skierowaną w dół i w tym kierunku przesuwa kursor. To, że klawisz **h** przesuwa kursor w lewo, a **l** w prawo nie powinno być niespodzianką, a góra pozostaje dla klawisza **k**. Klawisze te (jak i większość opisanych dalej) można poprzedzić liczbą. Przykładowo naciśnięcie kolejno **5**, **h** – przesunie kursor o **5** znaków w lewo, a **4**, **k** – o **4** wiersze w górę (dalej będzie używany skrótowy zapis **5h** albo **4k**).

Klawisz **H** przesuwa kursor na pierwszy wiersz ekranu, a **L** na wiersz ostatni. Skojarzenie z *High* i *Low* jest też jak najbardziej na miejscu. Dodatkowo klawisz **M** (*Middle*) umieszcza kursor w środkowym wierszu ekranu.

Znając konwencję stosowaną w wyrażeniach regularnych nie powinno dziwić, że klawisz ^ (daszek, *caret*) przesuwa kursor na początek wiersza, a klawisz \$ (dolar) na koniec wiersza (zamiast ^ można nacisnąć **0** (zero) – działa tak samo). Ponadto **b** (*back*) na początek poprzedniego słowa, **w** (*word*) na początek następnego słowa. Klawisze **Ctrl+f** (*forward*), **Ctrl+b** (*backward*) przewijają tekst odpowiednio o ekran do przodu albo o ekran do tyłu.

Bardzo przydatny jest klawisz % (procent) – wyszukuje nawias będący parą dla nawiasu wskazywanego przez kursor.

Wyszukiwanie tekstu działa dokładnie tak samo jak to zostało opisane przy programie less. Klawisze / i ? umożliwiają zainicjowanie poszukiwania w przód albo wstecz, a klawisze \mathbf{n} i \mathbf{N} wyszukują następne wystąpienie też odpowiednio w przód i wstecz.

Bez przechodzenia do trybu edycji można klawiszem \mathbf{x} skasować znak wskazywany przez kursor, a – klawiszem \mathbf{X} znak poprzedzający kursor.

Po naciśnięciu klawisza **d** (*delete*) zostanie skasowany tekst określony przez następujące po nim przesunięcie kursora. To znaczy, że jeżeli następnym klawiszem będzie **h** albo **l**, to zostanie skasowany znak na lewo albo na prawo (pozycję kursora określa jego lewa krawędź). Jeżeli będzie to klawisz **b** albo **w** to zostanie skasowany fragment od kursora do początku bieżącego albo następnego słowa, a jeżeli ^ albo \$, to odpowiednio zostanie skasowany fragment wiersza występujący przed albo za kursorem.

Fragment wiersza od kursora do końca linii można również skasować naciskając **D** (zamiast **d\$**).

Naciśnięcie dd (dwa razy d) skasuje cały wskazywany wiersz.

Polecenie **dd** można poprzedzić liczbą, wówczas przykładowo **3dd** oznacza skasuj wiersz aktualny i dwa następne. Podobnie **2db** oznacza skasuj do początku drugiego słowa w lewo, a **4dl** skasuj 4 następne znaki (wliczając wskazywany przez kursor).

Kasowany tekst trafia do bufora (schowka). Można go wstawić w miejscu wskazywanym przez kursor klawiszem \mathbf{p} albo \mathbf{P} (*paste*). Jeżeli w buforze jest fragment wiersza, to zostanie wstawiony w bieżącej linii przed (\mathbf{P}) albo za (\mathbf{p}) kursorem, jeżeli w buforze jest jeden lub więcej wierszy, to zostanie on (lub one) wstawiony przed (\mathbf{P}) albo za (\mathbf{p}) bieżącym wierszem.

Tekst może być umieszczony w buforze bez kasowania po naciśnięciu klawisza **y** (*yank*) według tych samych reguł co w przypadku klawisza **d**, czyli zależnie od tego który z klawiszy **h**, **l**, **b**, **w**, ^, **\$** zostanie naciśnięty po klawiszu **y**. Dwukrotne naciśnięcie klawisza **y** (**yy**) albo klawisza **Y** umieszcza w buforze cały bieżący wiersz. Podobnie jak w przypadku **dd** można użyć sekwencji typu **4yy**, która w tym przypadku umieści w buforze wiersz bieżący i trzy następne. Przez dalszą analogię do polecenia **d** można umieścić w buforze fragment do początku drugiego słowa w prawo **2yw**, albo znak pod kursorem i trzy poprzedzające **4yh**, albo wiersz aktualny i dwa poprzedzające **3yk**.

Dowolny wiersz tekstu można zaznaczyć naciskając klawisz \mathbf{m} (*mark*), a następnie dowolny inny klawisz z **literą**, która będzie nazwą znacznika. Ustawiony znacznik nie ma żadnej reprezentacji na monitorze. Po prostu

trzeba o nim pamiętać. Później, w dowolnej chwili naciśnięcie kolejno klawiszy ' (apostrof) **litera** przeniesie kursor do zaznaczonego wiersza.

Ta cecha umożliwia również utworzenie bloku tekstu. Po ustawieniu kursora w pierwszym wierszu bloku naciskamy klawisze **m**, **p**. Przemieszczamy kursor za ostatni wiersz bloku i naciskamy **m**, **o**. Naciskamy kolejno klawisze

 \mathbf{y} , ' (apostrof), \mathbf{p} , ' (apostrof), \mathbf{o}

i w buforze znajdzie się zaznaczony fragment tekstu (można było oczywiście użyć dowolnych innych nazw znaczników zamiast \mathbf{p} i \mathbf{o}). Teraz można wkleić bufor w dowolnym miejscu używając opisanych już wyżej poleceń (klawiszy) \mathbf{p} i \mathbf{P} .

Bloki mogą być również wykorzystywane przy usuwaniu (\mathbf{d}) i zmienianiu (\mathbf{c}) tekstu (polecenie \mathbf{c} zostanie opisane dalej).

Nieco podobnie do klawisza \mathbf{x} działa klawisz \mathbf{r} (*replace*). Edytor przechodzi do trybu edycji, ale tylko do czasu naciśnięcia następnego klawisza. W rezultacie po naciśnięciu klawisza \mathbf{r} znak wskazywany przez kursor zostanie zastąpiony znakiem następnego naciśniętego klawisza.

Po naciśnięciu klawisza \mathbf{R} edytor przechodzi w tryb zastępowania, aż do czasu naciśnięcia klawisza \mathbf{Esc} (wyjścia z trybu edycji).

Inne klawisze powodujące przejście w tryb edycji to:

- $\mathbf{a}~-(append)$ wstawianie za aktualną pozycją kursora,
- A wstawianie na końcu bieżącego wiersza,
- i (insert) wstawianie przed aktualną pozycją kursora,
- ${f I}~-$ wstawianie na początku bieżącego wiersza,
- s (substitute) podobnie do r zastąpi wskazywany znak znakiem naciśniętego klawisza, ale pozostanie w trybie wstawiania,
- ${\bf S}\,$ skasuje bieżącą linię i pozostanie w trybie wstawiania na początku pustej linii,

Wspomniany klawisz \mathbf{c} (*change*), podobnie do \mathbf{d} i \mathbf{y} , zależnie od naciśniętego następnego klawisza **hlbw^\$** umożliwi zastąpienie:

- \mathbf{h} poprzedniego znaku,
- $l \quad \ następnego \ znaku,$
- **b** fragmentu od początku bieżącego słowa,
- ${\bf w}~$ fragmentu do początku następnego słowa,
- ^ fragmentu od początku wiersza,
- **\$** fragmentu do końca wiersza.

Po zakończeniu zamiany przejdzie w tryb wstawiania.

Zamianę do końca wiersza można łatwiej osiągnąć klawiszem \mathbf{C} . Polecenia \mathbf{s}, \mathbf{S} i \mathbf{c} podobnie do \mathbf{d} i \mathbf{y} można poprzedzić liczbą. Dla ułatwienia polecenia wywołane klawiszami s, c i C obrazują zasięg zastępowania znakiem

Klawisze o, O utworzą nową, pustą linię odpowiednio poniżej lub powyżej bieżącej.

Po naciśnięciu klawisza \mathbf{J} (*join*) do końca wiersza wskazywanego przez kursor zostanie dołączony wiersz następny (dwa wiersze zostaną połączone w jeden).

Klawisze $\langle i \rangle$ zmieniają wielkość wcięcia. Naciśnięte dwukrotnie ($\langle < albo \rangle >$) zmieniają wcięcie aktualnego wiersza. Można je poprzedzić liczbą, wówczas zostanie zwiększone albo zmniejszone wcięcie zadanej liczby kolejnych wierszy.

Oczywiście działają kombinacje typu 2 < k oznaczające zmniejsz wcięcie wiersza bieżącego i dwóch poprzedzających.

Kolejno naciśnięte klawisze < (albo >) '(apostrof) litera, gdzie litera jest nazwą znacznika, zmieniają wcięcie fragmentu tekstu od bieżącej pozycji kursora do wiersza zaznaczonego literą.

Klawisz u (undo) cofa ostatnią operację. Niestety tylko ostatnią.

Naciśnięcie klawiszy $\mathbf{Ctrl} + \mathbf{G}$ wyświetli nazwę bieżącego pliku, jego wielkość oraz lokalizację kursora.

Aby możliwe było wydanie polecenia, może zachodzić potrzeba wyjścia z trybu edycji poprzez naciśnięcie klawisza **Esc**. Wszystkie poniższe polecenia rozpoczynają się od znaku : (dwukropka) i kończy je naciśnięcie klawisza **Enter**.

- :liczba przechodzi do wiersza wskazanego przez liczba,
- :set ai włącza utrzymywanie wcięcia z poprzedniego wiersza,
- :set noai wyłącza utrzymywanie wcięcia z poprzedniego wiersza,
- :set backup przed zapisaniem bieżącej wersji pliku nazwa poprzedniej zostanie zmieniona przez dodanie na końcu znaku ~ (tylda),
- :set nobackup nie będzie zachowywał poprzedniej wersji pliku,
- :set t
s=n powoduje, że naciśnięcie klawisza Tab przesunie kursor do następnej kolumny o numerze podzielnym prze
z ${\bf n},$
- :set sw=n ustawia szerokość (liczbę znaków n), o jaką klawisze <
 i > zmieniają wielkość wcięcia,
- :w zapisuje zmiany wprowadzone w tekście,
- :w nowa_nazwa zapisuje plik pod nową nazwą (pomiędzy w, a nowa_nazwa jest spacja),
- :
q wyjdzie z edytora, o ile nie ma konieczności zapisania zmian,
- :wq zapisze plik i wyjdzie z edytora,
- :q! wyjdzie z edytora bez zapisywania zmian,
- :
x wyjdzie z edytora jeżeli to potrzebne wcześniej zapisze zmiany,

dokładnie ten sam efekt można osiągnąć naciskając klawisze \mathbb{ZZ} (dwukrotnie wielkie \mathbb{Z} bez poprzedzającego dwukropka),

— :e nazwa_pliku – otwarcie nowego pliku do edycji (pomiędzy e, a nazwa_pliku jest spacja).

Jak widać obsługa edytora vi wymaga na początku pewnego wysiłku, a przecież nie zostały opisane wszystkie jego możliwości. Trzeba pamięciowo opanować znaczny zasób wiedzy. Stąd też edytor ma sporą grupę przeciwników. Jednak ci, którzy pokonają pierwsze trudności twierdzą, że jest to najlepszy edytor na świecie, (no chyba że akurat za taki uważają emacs'a).

5.3.3. Edytor nano

Znacznie prostszym, bardziej intuicyjnym w obsłudze i niemal równie często występującym w różnych dystrybucjach systemu GNU/Linux jest edytor **nano**.

Jego pierwowzorem jest edytor pico, na który niestety są nałożone ograniczenia licencyjne. nano powstał jako wolny klon pico dostępny na zasadach licencji GPL. Program został później bardzo rozbudowany, a obecnie jego nazwa nano $(nano = 10^{-9})$ może sugerować olbrzymie możliwości w stosunku do pico $(pico = 10^{-12})$ (a może jest to tylko gra słów). nano ma skromniejsze możliwości od vi, ale brakujące funkcje nie należą do najważniejszych i w codziennej pracy można obyć się bez nich.

Sposoby uruchomienia edytora nano są analogiczne do vi:

nano

nano nazwa_pliku

```
nano +numer_wiersza nazwa_pliku
```

jednak samodzielny + (plus) jako znane z \mathtt{vi} polecenie przejścia na koniec pliku nie jest rozpoznawany.

Inne użyteczne opcje to:

- -B przed zachowaniem bieżącej wersji dokumentu zmieni nazwę poprzedniej dodając na końcu ~ (tyldę), tę cechę można na bieżącą przełączać klawiszami **M-B** (jak rozumieć zapisy typu **M-B** i **^K** zostanie wyjaśnione za chwilę),
- -E edytor będzie automatycznie zamieniał znaki tabulacji na odpowiednią liczbę spacji, tę cechę można na bieżąco przełączać klawiszami M-Q,
- -i ustawia tryb automatycznego utrzymywania wcięcia z poprzedniego wiersza, tę cechę można na bieżąco przełączać klawiszami M-I,
- $-{\bf k}$ normalnie naciśnięcie klawiszy ${\bf ^K}$ powoduje skasowanie całego wiersza, w przypadku użycia opcji $-{\bf k}$ wiersz będzie kasowany od pozycji kursora do końca,

- -m włącza elementarną obsługę myszki, tę cechę można na bieżąco przełączać klawiszami M-M,
- -T n ustawia szerokość tabulacji i wcięć na n znaków (domyślnie 8).

W odróżnieniu od vi, nano pracuje stale w trybie edycji, a odpowiednie polecenia wydaje się kombinacjami klawisza **Ctrl** albo **Alt** z innymi klawiszami. Klawisze lewy i prawy **Alt** są rozróżniane, zgodnie z oczekiwaniami powinien działać lewy.

Po uruchomieniu w najwyższym wierszu ekranu wyświetlana jest wersja edytora, nazwa pliku i ewentualnie informacja o potrzebie zapisania pliku.

W dwóch dolnych wierszach ekranu wyświetlone są najczęściej używane funkcje. Przyjęta jest następująca konwencja: **X** oznacza równoczesne naciśnięcie klawisza **Ctrl** i klawisza **X**, **M-X** oznacza równoczesne naciśnięcie klawisza **Alt** i klawisza **X**. Jeżeli kombinacja **Alt+X** nie działa prawidłowo, można nacisnąć kolejno klawisze **Esc**, a następnie **X**.

Trzeci wiersz od dołu to wiersz statusu, komunikatów i dialogów.

Po opanowaniu podstawowych opcji można wyłączyć pomoc w ostatnich dwóch wierszach klawiszami **M-X**. Wiersz statusu przesunie się wówczas o dwa wiersze w dół udostępniając dodatkowy obszar na edycję pliku. Tymi samymi klawiszami w każdej chwili można przywrócić pomoc w dwóch ostatnich wierszach.

Drugi wiersz ekranu (poniżej tytułowego) normalnie pozostaje pusty. Aby wykorzystać go jako obszar edycyjny należy nacisnąć klawisze **M-O**.

Po naciśnięciu klawiszy $\mathbf{\hat{G}}$ zostanie wyświetlony pierwszy ekran pomocy edytora (rys. 5.3). Stosownie do nowej zawartości części głównej ekranu zmieni się zawartość dwóch najniższych wierszy. Aby opuścić system pomocy należy nacisnąć klawisze $\mathbf{\hat{X}}$. Ta sama kombinacja klawiszy kończy pracę edytora. W razie potrzeby edytor zapyta, czy zapisać plik po wprowadzeniu zmian.

Edytor **nano** obsługuje klawisze kursora, jednak ich działanie można osiągnąć również inaczej:

- \mathbf{P} (previous) albo \mathbf{B} (back) poprzedni znak,
- \mathbf{N} (next) albo \mathbf{F} (forward) następny znak,
- $\mathbf{\hat{Y}}$ poprzednia strona,
- \mathbf{V} następna strona,
- \mathbf{A} początek wiersza
- **^E** koniec wiersza,
- **^spacja** następny wyraz,
- M-spacja poprzedni wyraz,
- M- $\$ albo M-| początek pliku,
- M-/ albo M-? koniec pliku.

Oprócz sterowania kursorem ważne są następujące polecenia:
| (| GNU nano 2.0 | .7 New Buffer | • | | | | |
|---|--------------|--|---|--|--|--|--|
| The notation for shortcuts is as follows: Control-key sequences are notated with a caret (^) symbol and can be entered either by using the Control (Ctrl) key or pressing the Escape (Esc) key twice. Escape-key sequences are notated with the Meta (M-) symbol and can be entered using either the Esc, Alt, or Meta key depending on your keyboard setup. Also, pressing Esc twice and then typing a three-digit decimal number from 000 to 255 will enter the character with the corresponding value. The following keystrokes are available in the main editor window. Alternative keys are shown in parentheses: | | | | | | | |
| ^G | (F1) | Display this help text | | | | | |
| ^X | (F2) | Close the current file buffer / Exit from nano | | | | | |
| ^0 | (F3) | Write the current file to disk | | | | | |
| ^J | (F4) | Justify the current paragraph | | | | | |
| ^R | (F5) | Insert another file into the current one | | | | | |
| ^₩ | (F6) | Search for a string or a regular expression | | | | | |
| ^Y | (F7) | Move to the previous screen | | | | | |
| ^V | (F8) | Move to the next screen | | | | | |
| ^L | Refresh | ☆ Prev Page | ▲ | | | | |
| ^χ | Exit | | ▼ | | | | |

Rysunek 5.3: Ekran pomocy edytora nano

- M-G przejdź do wiersza o zadanym numerze, po przecinku można zadać również numer kolumny,
- $\mathbf{\hat{C}}$ pokazuje bieżącą pozycję kursora, klawisze $\mathbf{\hat{C}}$ przerywają również różnego rodzaju dialogi pytania zadawane przez edytor,
- ^K skasuj bieżący wiersz umieszczając go w buforze,
- M-T skasuj tekst od pozycji kursora do końca pliku umieszczając go w buforze,
- M-[^] albo M-6 kopiuj bieżący wiersz do bufora można w ten sposób umieścić w buforze kilka kolejnych wierszy,
- ^U wklej zawartość bufora w miejscu wskazywanym przez kursor,
- M-Q przełącza tryb konwersji tabulacji na spacje,
- M-I przełącza tryb automatycznego utrzymywania wcięcia z poprzedniego wiersza,
- M-} zwiększ wcięcie wiersza,
- M-{ zmniejsz wcięcie wiersza,
- **M-A** ustaw znacznik na bieżącej pozycji kursora, po przesunięciu kursora w inne miejsce obszar od znacznika do bieżącego położenia kursora stanowi blok. Na tym bloku można przeprowadzić operacje:
 - kopiowanie do bufora $M-^{,}$
 - wycięcie z umieszczeniem w buforze $\mathbf{\hat{K}},$
 - zmiana głębokości jego wcięcia **M-**} albo **M-**{.
 - Jeżeli jest ustawiony znacznik, ponowne naciśnięcie M-A zdejmie go,
- M-M-jeżeli jest to możliwe włącza (albo wyłącza) elementarną obsługę myszki, kliknięcie przemieszcza kursor, ponowne kliknięcie w tej samej

pozycji ustawia znacznik (tak jak **M-A**), trzecie kliknięcie w tej samej pozycji zdejmuje znacznik,

- M-D wyświetla informacje o pliku (liczbę słów, wierszy, znaków), a jeżeli jest zaznaczony blok, to zamiennie wyświetla te same informacje o bloku,
- M-] znajdź nawias będący parą dla nawiasu wskazywanego przez kursor,
- **^W** szuka łańcucha znaków lub wyrażenia regularnego,
- ${\bf M}\text{-}{\bf W}$ powtórz ostatnie wyszukiwanie,
- M-R zastąp łańcuch znaków lub wyrażenie regularne,
- ^R wstaw do bieżącego pliku zawartość innego pliku,
- ^O zapisuje bieżący plik (umożliwia zmianę nazwy), pozostaje w trybie edycji,
- $\mathbf{\hat{X}}$ jeżeli plik został zmodyfikowany, to pyta o pozwolenie na zapis i postępuje zgodnie z odpowiedzią, a następnie kończy pracę edytora.

Jak widać możliwości edytora są więcej niż wystarczające i może nam brakować tylko poszukiwania wstecz i funkcji undo / redo. Ta druga funkcja jest zapowiadana w następnych wersjach edytora.

5.3.4. Edytor mcedit

Ostatnim edytorem jaki zostanie omówiony jest mcedit. Jest to edytor wbudowany rozbudowanego zarządcy plików o nazwie mc.

Podobnie do wcześniej omówionych edytorów mcedit uruchamia jedno z trzech poleceń:

```
mcedit
mcedit nazwa_pliku
mcedit nazwa_pliku +numer_wiersza
```

Trzeci wariant wymaga odwrotnej kolejności argumentów w stosunku do nano. vi nie jest wrażliwy na kolejność tych argumentów.

Jeszcze jedną przydatną opcją uruchamiania edytora może okazać się –b – edytor rozpocznie pracę w trybie monochromatycznym.

Fizycznie rzecz biorąc nie ma edytora (wykonywalnego pliku dyskowego) mcedit jako takiego. Jest on wkompilowany do programu mc, a mcedit jest tylko linkiem symbolicznym do mc. Uruchomienie programu mc z opcją -e uruchomi go w trybie edytora.

mcedit jest zaadaptowaną terminalową wersją edytora o nazwie cooledit. Zdaniem autorów określenie *cool* jest tutaj jak najbardziej na miejscu. Jest to naprawdę *świetny* edytor. A teraz konkrety.

Po uruchomieniu w najwyższym wierszu ekranu zostaną wyświetlone:

- nazwa pliku,
- flagi statusu w nawiasach kwadratowych,
- pozycja w pliku określona przez kolumnę, wiersz i offset,
- rozmiar pliku wyrażony ilością wierszy i ilością znaków,
- znak wskazywany przez kursor oraz wartość odpowiadającego mu bajtu wyrażona dziesiętnie i szesnastkowo.

| kolo c | [] $0 + []$ $0 + []$ $16/126] *(1126/2779b) - * 42 0 × 24$ | | | | | | | |
|------------|--|--------|--|--|--|--|--|--|
| */ | [] 0 L.[10+ 0 10/130] (1120/3/780)= 42 0X2A | | | | | | | |
| <i>'</i> | Pomoc | | | | | | | |
| #includ | Wbudowany edytor plików ma wiekszość funkcii posiadanych | | | | | | | |
| #includ | przez inne edytory pełno-ekranowe. Jest wywoływany po | | | | | | | |
| #includ | naciśnieciu klawisza F4 o ile opcja use internal edit jest | | | | | | | |
| #includ | ustawiona w pliku startowyn. Ma maksymalny rozmiar pliku | | | | | | | |
| #includ | wynoszący szesnaście megabajtów i potrafi bez skazy | | | | | | | |
| #includ | edytować pliki binarne. | | | | | | | |
| #includ | | | | | | | | |
| | Opcje, które aktualnie posiada to: kopiowanie, | | | | | | | |
| #define | przenoszenie, kasowanie, wycinanie i wklejanie blokow; | | | | | | | |
| tupedef | klawisz dla klawisza undo; rozciągane menu; wklejanie | | | | | | | |
| int | roqularpych (i paszo właspo funkcio scanf-printf szukajaco | | | | | | | |
| cha | i zamioniajaco): strząłki z Sbiftom zazpaczająco toksty w | | | | | | | |
| } shms | stylu MSW-MAC (tylko dla konsoli typu Linux): przełaczanie | | | | | | | |
| , <u> </u> | trybu wstawiania-zastepowania: opcja pozwalajaca na | | | | | | | |
| shms t* | "przerzucenie" bloku tekstu przez komendę powłoki jak na | | | | | | | |
| _ | przykład indent. | | | | | | | |
| int mai | | | | | | | | |
| | | | | | | | | |
| pid | | | | | | | | |
| Pomoc | 2Indeks 3Poprz 4 5 6 7 8 9 10K | ończ 👻 | | | | | | |

Rysunek 5.4: Ekran pomocy edytora mcedit w trybie monochromatycznym

Flagi statusu mogą obrazować:

- ${\bf M}$ plik został zmodyfikowany i wymaga zapisu,
- B zaznaczony jest blok zwykły,
- \mathbf{C} zaznaczony jest blok kolumnowy,
- ${\bf R}~$ edytor rejestruje naciskane klawisze jako tak zwane makro.
- ${\bf O}$ edytor jest w trybie nadpisywania stan ten można przełączać klawiszem ${\bf Insert}.$

Klawisze kursora są obsługiwane intuicyjnie. Dodatkowe funkcje:

- Ctrl+Home na początek pliku,
- Ctrl+End na koniec pliku,
- Ctrl+ \leftarrow poprzednie słowo,
- Ctrl+ \rightarrow następne słowo.

W najniższym wierszu ekranu znajduje się pomoc opisująca klawisze funkcyjne:

- $\mathbf{F1}$ wyświetla ekran pomocy (rys. 5.4),
- F2 zapisuje plik na dysku,
- F3 ustawia znaczniki początku i końca bloku,
- F4 wywołuje okno dialogowe "Zastąp" (rys. 5.8),

- F5 kopiuje blok uprzednio zaznaczony naciśnięciami klawisza F3 w miejsce wskazywane przez kursor,
- F6 przenosi blok w miejsce wskazywane przez kursor,
- F7 wywołuje okno dialogowe "Szukaj",
- F8 usuwa zaznaczony blok, jeżeli blok nie jest zaznaczony, to usuwa bieżący wiersz,
- $\mathbf{F9}$ uaktywnia menu,
- F10 kończy pracę edytora, w razie potrzeby pyta czy zapisać zmodyfikowany plik.

Edytor może być wykorzystywany nawet na klawiaturach nie posiadających klawiszy funkcyjnych, a także w przypadkach nietypowej emulacji terminala (gdy klawisze funkcyjne generują nierozpoznawane kody) lub gdy klawisze funkcyjne są przechwytywane przez program terminala (konsoli). Należy wówczas posłużyć się sekwencją **Esc cyfra**, gdzie **cyfra** jest odpowiednim (**1**.. **9**, **0**) klawiszem z bloku alfanumerycznego klawiatury. Zamiast **F10** można dwukrotnie nacisnąć klawisz **Esc** albo **Ctrl+C**.

Edytor może mieć wkompilowaną obsługę myszki. Wówczas kliknięcie w najwyższy, tytułowy wiersz ekranu uaktywni menu (tak jak naciśnięcie klawisza **F9**). Możliwa jest obsługa myszką zarówno menu głównego, jak też dolnego, obsługiwanego normalnie klawiszami funkcyjnymi. Przeciągnięcie myszką po edytowanym tekście zaznaczy blok podobnie do klawiszy **F3**.

Jeżeli edytor uruchomiony jest w konsoli graficznej, to myszkę można obsługiwać przy naciśniętym klawiszu **Shift**. Można w ten sposób zaznaczyć blok oraz prawym klawiszem myszy wywołać menu kontekstowe (rys. 5.5), które umożliwi skopiowanie bloku do schowka systemowego i przeniesienie go do innych programów.

Możliwe jest również wklejenie bloku ze schowka systemowego do programu **mcedit**. Tę operację inicjuje wybranie odpowiedniej opcji z menu kontekstowego, wywoływanego prawym klawiszem myszy przy naciśniętym klawiszu **Shift** (rys. 5.5). Ponieważ odbywa się ona przez bufor klawiatury, pożądane jest wyłączenie na ten czas automatycznego utrzymywania wcięć. Jak to zrobić pokazuje rysunek 5.6.

Niektóre klawisze funkcyjne mogą być użyte razem z klawiszem **Shift**, zmienia się wówczas ich znaczenie:

- Shift+F1 wywołuje "menu użytkownika" zależne od typu pliku (menu może być opisane na przykład w pliku /usr/share/mc/cedit.menu),
- Shift+F2 umożliwia zapisanie pliku pod zmienioną nazwą,
- Shift+F3 ustawia znaczniki początku i końca bloku kolumnowego,
- Shift+F4 powtarza operację "Zastąp" zadaną klawiszem F4,



Rysunek 5.5: Menu kontekstowe

- Shift+F5 wstawia blok z pliku (domyślnie ~/.mc/cedit/cooledt.clip),
- Shift+F7 powtarza wyszukiwanie zadane klawiszem F7,
- Shift+F9 wywołuje skrypt ~/.mc/cedit/edit.indent.rc umożliwia to wykorzystanie zewnętrznego narzędzia formatującego tekst.

Ważniejsze klawisze sterujące:

- **^F** zapisuje blok w pliku (domyślnie ~/.mc/cedit/cooledt.clip),
- ^K usuwa fragment wiersza od kursora do końca,
- ^N zamyka bieżący plik i rozpoczyna pracę z nowym,
- $^{\mathbf{O}}$ wywołuje podpowłokę, ponowne naciśnięcie $^{\mathbf{O}}$ powraca do edytora,
- ^R rozpoczyna i kończy rejestrowanie makro,
- $\mathbf{^{-}U}$ wielopoziomowe undo, cofa ostatnio wykonane operacje,
- **^Y** usuwa bieżący wiersz,
- $\mathbf{M}\textbf{-}\mathbf{B}$ znajdź nawias będący parą do nawiasu wskazywanego przez kursor,
- \mathbf{M} - \mathbf{L} idź do wiersza,
- $\mathbf{M}\textbf{-}\mathbf{U}$ wywołuje zewnętrzne polecenie i wstawia wyniki jego działania do edytowanego tekstu.

Większość z tych skrótów klawiaturowych ma swoje odpowiedniki w menu.

Poprzez system menu można również skonfigurować edytor. Ilustruje to zrzut ekranu na rysunkach 5.6 i 5.7:



Rysunek 5.6: Konfiguracja mcedit



Rysunek 5.7: Tryby zapisu mcedit

Wiele opcji ma charakter uznaniowy i zależy wyłącznie od preferencji użytkownika, ale autorzy sugerują aby:

- ustawić wypełnianie tabulacji spacjami,
- szerokość tabulacji ustawić na 4,
- w większości wypadków należy korzystać z automatycznych wcięć,
- trybu zapisu nie ustawiać na szybki (rys. 5.7).

Edytor ma kilka cech bardzo ułatwiających pisanie programów. Oto niektóre z nich:

- wiersz podpowiedzi i system menu umożliwiają rozpoczęcie pracy praktycznie bez wcześniejszego przygotowania,
- ma estetycznie dobrany zestaw kolorów, ale może również pracować w trybie monochromatycznym,
- po rozszerzeniu pliku rozpoznaje język programowania i odpowiednio koloruje składnię,
- jeżeli kursor wskazuje jeden z nawiasów, to nawias będący parą do niego zmienia kolor czcionki,
- wywołanie podpowłoki klawiszami **^O** umożliwia skompilowanie i uruchomienie wersji rozwojowej programu, następne naciśnięcie klawiszy **^O** – wraca do edytora i umożliwia wygodną kontynuację pracy,
- polecenia wyszukiwania $\mathbf{F7}$ i zamiany $\mathbf{F4}$ obsługują wyrażenia regularne,
- umożliwia przenoszenie kawałków kodu pomiędzy plikami poprzez zewnętrzny bufor (schowek) w postaci pliku,
- obsługuje blok kolumnowy,
- obsługuje makra,
- przy odpowiednio skonfigurowanym systemie można z niego wysłać maila.

Oczywiście większość tych cech (albo cechy równoważne) miały również edytory omawiane wcześniej, ale w mcedit są one udostępnione w sposób wyjątkowo wygodny, przyjazny, sympatyczny – po prostu *cool*.

Jedyną wadą jaką autorzy dostrzegają w tym edytorze jest brak możliwości zmiany wielkości wcięcia bloku. Stosunkowo łatwo można tę niedogodność obejść i to na kilka sposobów. Poniżej zostaną omówione trzy z nich, a przy tej okazji zostanie pokazane również:

— jak operować na znakach sterujących,

- czym są makra i jak się nimi się posługiwać,
- do czego może służyć blok kolumnowy.

Dla poprawnego powtórzenia przykładów trzeba skonfigurować edytor na wypełnianie tabulacji spacjami.

Jak wspomniano wcześniej wiersz pliku tekstowego ograniczony jest znakiem końca (wysuwu) wiersza **LF**. Jeżeli potraktować go jako bajt, to będzie on miał wartość **10** (**0xA** szesnastkowo). Aby zwiększyć wcięcie następnego wiersza można zastąpić znak końca wiersza przez znak końca wiersza i 4 spacje. Jak to osiągnąć?

- należy nacisnąć klawisz
 ${\bf F4}$ aby wywołać okno dialogowe "Zastąp" (rys. 5.8),
- w polu "W
prowadź poszukiwany napis" naciskamy kolejno klawisze
 $\mathbf{\hat{Q}},$ $\mathbf{\hat{J}}.$

Wyjaśnienie: pierwszy klawisz $\mathbf{\hat{Q}}$ jest sygnałem, że następny klawisz ma być potraktowany literalnie, bez żadnej interpretacji. Litera **J** jest dziesiątą literą w alfabecie łacińskim. Klawisz $\mathbf{\hat{J}}$ wstawia więc znak o kodzie 10, czyli znak końca wiersza. Znak o kodzie 10 nie ma specjalnej reprezentacji graficznej, dlatego zastępczo w jego miejscu pojawi się (kropka).

| Zastąp Wprowadź poszukiwany napis: Wprowadź zastępujący napis: (*) Wprowadź kolejność zastępowania argumentów, np.: 3,2,1, [] Rozróżnianie wielkości liter [] Tylko całe słowa [x] Potwierdzanie [] Jylko całe słowa [x] Potwierdzanie [] Wyrażenie regularne [] Zastąp wszystkie [] Wyrażenie "scanf" [< OK >] [Anuluj] | | | | | |
|---|--|--|--|--|--|
| tinclude «sye/types.h» | | | | | |
| Potwierdzanie | | | | | |
| int rb, qi; char buff[BUFSIZE]; | | | | | |

Rysunek 5.8: Wcinanie tekstu przez zastępowanie LF

- w polu "Wprowadź zastępujący napis" należy nacisnąć kolejno \mathbf{Q} , \mathbf{J} i cztery razy klawisz **spacji**,
- naciśnięcie klawisza Enter rozpocznie proces zastępowania,
- w oknie dialogowym "Potwierdzanie" należy wybrać "Zastąp" (rys. 5.8),
- należy naciskać tak długo klawisze Shift+F4, aż zostanie uzyskana zmiana wcięcia dla odpowiedniego fragmentu tekstu.

Jak łatwo się domyślić zmniejszanie wcięcia tą metodą będzie polegało na zastąpieniu znaku zmiany wiersza i czterech spacji przez sam znak zmiany wiersza.

Jeżeli do zmiany wielkości wcięcia są stosowane znaki tabulacji to można je wpisać jako tekst zastępowany lub zastępujący naciskając kolejno $\mathbf{\hat{Q}}$, **Tab** albo $\mathbf{\hat{Q}}$, $\mathbf{\hat{I}}$.

Prostą i skuteczną metodą zmiany wielkości wcięcia może być posłużenie się **makrami**. Zastanówmy się jakie klawisze i w jakiej kolejności naciśnięte z pewnością spowodują zwiększenie wcięcia? **Home Spacja Spacja Spacja**. Dodajmy jeszcze powrót na początek wiersza **Home** i przejście do następnego \downarrow (strzałką w dół).

Procedura rejestracji makra wygląda następująco:

- naciskamy klawisze $\mathbf{\hat{R}}$ (pojawia się flaga rejestracji makra R),
- naciskamy kolejno 6 wymienionych klawiszy,
- kończymy rejestrację makra klawiszami \mathbf{R} , edytor pyta o nazwę makra niech będzie to litera I (*indent* wcięcie).

Od tej chwili naciśnięcie klawiszy **Alt+I** spowoduje wykonanie makra – czyli wcięcie wiersza. Klawisze **Alt+I** można przytrzymać tak długo, aż zostanie wcięta potrzebna liczba wierszy.

Makro zmniejszające wcięcie może wyglądać następująco: Home 4 razy \rightarrow (strzałka w prawo), F3 jako początek bloku, Home, F3 jako koniec bloku, F8 usunięcie bloku, \downarrow (strzałka w dół). Najwygodniej byłoby zarejestrować je pod nazwą U (unindent). Niestety Alt+U jest już zarezerwowane – wywołuje polecenie zewnętrzne i wstawia do edytowanego tekstu wyniki jego działania. Można wykorzystać sąsiadujące, nie zarezerwowane Y. W istocie, w nazwach makr małe i wielkie litery są rozróżniane, więc makro zwiększające wcięcie można nazwać i, a zmniejszające I.

Jeżeli zachodzi potrzeba powiększenia wcięcia tekstu, który jest już wcięty (na przykład z 4 na 8 znaków) można skorzystać z **bloku kolumnowego**:

- ustawiamy kursor o jeden znak na prawo od prawego dolnego rogu obszaru, dla którego należy powiększyć wcięcie,
- naciskamy klawisze **Shift+F3** (pojawi się flaga bloku kolumnowego **C**),
- przemieszczamy kursor w lewo o wymaganą liczbę znaków i w górę o wymaganą liczbę wierszy,
- naciskamy klawisz Shift+F3 aby zamknąć blok kolumnowy,
- naciskamy klawisz
 ${\bf F5}$ w celu skopiowania bloku kolumnowego i powiększenia w
cięcia.

Aby metodą bloku kolumnowego zmniejszyć wcięcie, po zaznaczeniu odpowiedniego obszaru należy nacisnąć $\mathbf{F8}$, co spowoduje skasowanie bloku i zmniejszenie wcięcia.

Formatowanie kodu jest możliwe również przy wykorzystaniu zewnętrznych narzędzi wywoływanych klawiszami **Shift+F9**. Omawianie tego zagadnienia wykracza jednak poza ramy niniejszego skryptu.



Rysunek 5.9: Blok kolumnowy

5.3.5. Podsumowanie

Przy okazji opisu edytorów zwrócono uwagę na te cechy, które decydują czy edytor będzie dobrym narzędziem dla programisty. Z drugiej strony patrząc, programista powinien umieć z tych cech korzystać aby ułatwić sobie pracę nad kodem źródłowym i spowodować by był on czytelny i przejrzysty. Program pisany bałaganiarsko, niestarannie jest trudny w utrzymaniu, wyszukiwaniu błędów i zniechęca do jego rozwoju.

Rozdział 6

Skrypty powłoki **bash**

| 6.1. | Wstęp |
|------|--|
| 6.2. | Uruchamianie skryptów powłoki 106 |
| 6.3. | Komentarze |
| 6.4. | Zmienne i argumenty wywołania skryptu 107 |
| 6.5. | Operacje arytmetyczne 108 |
| 6.6. | Operacje logiczne 108 |
| 6.7. | Sterowanie przebiegiem wykonania skryptu 110 |
| | 6.7.1. Instrukcja warunkowa if |
| | 6.7.2. Instrukcja wyboru case |
| | 6.7.3. Instrukcja iteracyjna while 112 |
| | 6.7.4. Instrukcja iteracyjna until |
| | 6.7.5. Instrukcja iteracyjna for in 114 |
| | 6.7.6. Instrukcja iteracyjna for |
| 6.8. | Pliki inicjalizacyjne powłoki bash |
| 6.9. | Podsumowanie |

6.1. Wstęp

W kilku wcześniejszych rozdziałach do wykonywania instrukcji systemu operacyjnego i wywoływania programów wykorzystywany, był interpreter poleceń **bash**, nazywany również powłoką. Powłoka **bash** ma jednak daleko większe możliwości. Możliwe jest tworzenie rozbudowanych programów interpretowanych przez powłokę, nazywanych zazwyczaj skryptami. Zagadnienie programowania skryptów jest bardzo rozległe i jako całość wykracza poza ramy niniejszego opracowania. W tym rozdziale zostaną przekazane jedynie podstawowe informacje, pokazujące jak rozpocząć programowanie skryptów powłoki.

Literatura dotycząca programowania skryptów powłoki jest bogata i łatwo dostępna, zarówno w formie elektronicznej jak drukowanej. Jeżeli czytelnik uzna zawarte poniżej informacje za niewystarczające, może skorzystać na przykład z książki "Programowanie w Systemie Linux" [26] albo kursu "Programowanie w Shellu: Bash" [27]

6.2. Uruchamianie skryptów powłoki

Skrypt powłoki jest zbiorem instrukcji rozpoznawanych przez interpreter bash, zapisanym w zwykłym pliku tekstowym. Przyjęte jest zapisywanie skryptów bash w plikach bez tzw. rozszerzeń.

Pierwszy, przykładowy skrypt będzie zawierał tylko jedną instrukcję:

echo "Hello World!"

Należy go zapisać w pliku o nazwie hw.

Jest kilka sposobów uruchomienia skryptu. Sposoby te nie są sobie równoważne, a różnice między nimi zostaną omówione dalej.

 $\verb+bash<+hw-można$ przekierować zawartość pliku hw na standardowe wejście interpretera $\verb+bash,$

bash hw – można uruchomić interpreter bash zadając mu nazwę pliku hw jako argument wywołania,

source hw – można zlecić wykonanie skryptu aktualnej powłoce poleceniem source,

.
hw – można zlecić wykonanie skryptu aktualnej powłoce poleceniem . (kropka).

Czwarty sposób wymaga modyfikacji skryptu:

```
#!/bin/bash
echo "Hello World!"
```

W pierwszym wierszu określono lokalizację interpretera **bash**, poprzedzoną znakami **#**! (jest to wymóg składniowy). Ponadto konieczne było nadanie plikowi hw atrybutu wykonywalności, na przykład poleceniem:

chmod +x hw

Od tej chwili skrypt może być uruchamiany poleceniem:

./hw

czyli tak jak każdy inny program.

Jedynie uruchomienie poleceniami **source** i . (kropka) uruchamia skrypt w aktualnej powłoce. Tak uruchomione skrypty mogą zmodyfikować zmienne środowiskowe aktualnej powłoki. Ponadto skrypty uruchomione w ten sposób nie powodują powstania kolejnego procesu, czyli zużywają mniej zasobów systemu operacyjnego.

6.3. Komentarze

Jeżeli użyty w przykładzie znak # (*hash*), nie występuje w połączeniu z ! (wykrzyknikiem) w pierwszym wierszu skryptu, oznacza to że tekst do końca wiersza jest komentarzem. Tekst ten zostanie oczywiście zignorowany podczas interpretacji skryptu.

#!/bin/bash

Komentarze w skrypcie Hello world! w bashu echo "Hello World!" #tutaj też można umieścić komentarz

6.4. Zmienne i argumenty wywołania skryptu

W skryptach powłoki można tworzyć zmienne i odwoływać się do nich. W tym zakresie obowiązują reguły opisane w sekcji traktującej o zmiennych środowiskowych.

W skrypcie można odwoływać się do argumentów wywołania skryptu. Wartość pierwszego argumentu przechowuje zmienna "\$1" i dalej konsekwentnie do "\$9".

Jeżeli do skryptu zostanie przekazane więcej niż 9 argumentów, możliwe jest skorzystanie z polecenia shift. W efekcie jego wywołania tracona jest zmienna "\$1", jej miejsce zajmuje zmienna "\$2". Zgodnie z tą regułą "przesuwane" są pozostałe zmienne, a miejsce zmiennej "\$9" zajmie niedostępna dotychczas dziesiąta zmienna.

Dodatkowo funkcjonują specjalne zmienne:

- "\$0" przechowuje nazwę skryptu,
- "\$#" przechowuje liczbę argumentów wywołania,
- "\$*" przechowuje wszystkie argumenty w postaci jednego napisu,
- "\$@" przechowuje wszystkie argumenty w postaci listy.

6.5. Operacje arytmetyczne

Powłoka bash potrafi wykonać podstawowe operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, modulo) na liczbach całkowitych. Wyrażenia arytmetyczne zapisywane są następująco: \$((wyrażenie)).

Przykładowy skrypt:

```
#!/bin/bash
x=11
y=7
echo "$x+$y="$(($x+$y))
echo "$x-$y="$(($x-$y))
echo "$x*$y="$(($x*$y))
echo "$x/$y="$(($x*$y))
echo "$x/$y="$(($x/$y))
echo "$x%$y="$(($x%$y))
```

wyświetli następujące wyniki:

```
11+7=18
11-7=4
11*7=77
11/7=1
11%7=4
```

6.6. Operacje logiczne

W stosunku do operacji arytmetycznych, **bash** potrafi wykonać bez porównania więcej operacji logicznych, które mogą być wykorzystane w konstrukcjach wyrażeń warunkowych.

Wyrażenia logiczne mogą być zapisywane na dwa równoważne sposoby:

```
test wyrazenie
```

[wyrazenie]

Może okazać się zaskakujące, że [(nawias kwadratowy otwierający) jest nazwą programu, który oczekuje] (nawiasu kwadratowego zamykającego) jako ostatniego argumentu wywołania. Zależnie od typu porównywanych wartości należy używać innych operatorów (opcji).

Dla danych napisowych można sprawdzić następujące warunki:

- test -z "napis" czy napis jest pusty,
- test -n "napis" czy napis nie jest pusty,
- test "napis_1" = "napis_2" czy napisy są identyczne,
- test "napis_1" != "napis_2" czy napisy są różne,
- test "napis_1" "<" "napis_2" czy napis_1 jest wcześniejszy leksykograficznie niż napis_2,
- test "napis_1" ">" "napis_2" czy napis_1 jest późniejszy leksykograficznie niż napis_2.

Dla danych liczbowych (całkowitych) można sprawdzić następujące warunki:

- test liczba_1 -gt liczba_2 czy liczba_1 jest większa od liczba_2 (greater than),
- test liczba_1 -lt liczba_2 czy liczba_1 jest mniejsza od liczba_2 (less than),
- test liczba_1 -ge liczba_2 czy liczba_1 nie jest mniejsza od liczba_2 (greater or equal),
- test liczba_1 -le liczba_2 czy liczba_1 nie jest większa od liczba_2 (less or equal),
- test liczba_1 -eq liczba_2 czy liczby są sobie równe (equal),
- test liczba_1 -ne liczba_2 czy liczby nie są sobie równe (not equal).

Istnieje bardzo rozbudowany zestaw testów dotyczących plików i ich atrybutów. Tutaj zostaną wymienione tylko te częściej stosowane:

- test -e plik czy plik istnieje,
- test -f plik czy plik jest zwykłym plikiem,
- test -d plik czy plik jest katalogiem,
- test -b plik, test -c plik czy plik jest odpowiednio urządzeniem blokowym, znakowym,
- test -L plik czy plik jest dowiązaniem symbolicznym,
- test -r plik, test -w plik, test -x plik czy aktualny użytkownik ma uprawnienia odpowiednio do odczytu, zapisu, uruchomienia pliku,
- -O plik czy aktualny użytkownik jest właścicielem pliku.

Wyrażenia warunkowe mogą być negowane operatorem ! (*not*) i łączone operatorami logicznymi -a (*and*) i -o (*or*). Przykłady:

- test "\$USER" = "student" -o -f hw prawdziwe, jeżeli zmienna środowiskowa \$USER przechowuje wartość student lub istnieje plik zwykły o nazwie hw (wystarczy, że spełniony jest jeden z warunków),
- test "\$#" -gt 0 -a "\$1" = "haslo" prawdziwe, jeżeli skrypt

został uruchomiony co najmniej z jednym argumentem i pierwszym z nich był napis haslo (muszą być spełnione oba warunki).

6.7. Sterowanie przebiegiem wykonania skryptu

6.7.1. Instrukcja warunkowa if

fi

Instrukcja warunkowa if ma 3 warianty składni: if polecenie then polecenia fi if polecenie then polecenia else polecenia fi if polecenie then polecenia elif polecenie then polecenia else polecenia

Po słowach kluczowych if i elif występuje polecenie. Może nim być instrukcja test opisana w poprzedniej sekcji, ale może to być również każde inne polecenie (program) ustawiające tzw. status zakończenia (kod wyjścia) programu.

Szczegółowe omawianie tematu statusu zakończenia programu wykracza poza ramy niniejszego skryptu. Wystarczy wiedzieć, że każdy program może poprzez ustawienie statusu swojego zakończenia zasygnalizować, czy jego wykonywanie zakończyło się poprawnie, czy nie.

W przypadku spełnionego warunku występującego po słowach kluczowych **if** i **elif** zostanie wykonana odpowiadająca im lista poleceń. Lista musi zawierać co najmniej jedno polecenie.

Jeżeli spełnione są warunki (dwa lub więcej), wówczas wykonywana jest wyłącznie lista poleceń, wystąpująca pod pierwszym spełnionym warunkiem.

Jeżeli nie jest spełniony żaden warunek występujący po słowach kluczowych if i elif, wówczas wykonana zostanie lista poleceń występująca po słowie kluczowym else – jeżeli takie występuje.

Całość konstrukcji if elif else kończy słowo kluczowe fi.

Prosty przykład ilustrujący przedstawione informacje:

```
#!/bin/bash
#skrypt należy uruchomić podając jako argument
#nazwę pliku źródłowego bez kończącego .c
if gcc -o "$1" "$1".c
then
    echo "kompilacja zakończona sukcesem"
    echo "uruchamiam program"
    ./"$1"
else
    echo "kompilacja zakończyła się klęską"
fi
```

W szczególnych przypadkach całą konstrukcję można zapisać w jednym wierszu. Uproszczony przykład:

if gcc \$1.c; then ./\$1; fi

należy zwrócić uwagę na ; (średniki).

6.7.2. Instrukcja wyboru case

Składnia instrukcji wyboru case jest następująca:

```
case napis in
wzor_1)
polecenia
;;
wzor_2)
polecenia
;;
*)
polecenia
;;
esac
```

Instrukcja warunkowa if umożliwiała testowanie dowolnych warunków. Jeżeli zachodzi tylko potrzeba dopasowania napisu do jednego z wielu wzorców, można posłużyć się instrukcją wyboru case. Po dokładnym omówieniu instrukcji if, w przypadku instrukcji case wystarczy prosty przykład:

```
#!/bin/bash
case "$USER" in
  "student")
    echo "Cześć Student"
    ;;
    "root")
    echo "Witaj Root"
    ;;
    "*")
    echo "Jeszcze Ciebie nie znam, drogi $USER"
    ;;
esac
```

Podobnie jak dla instrukcji **if**, wykonana zostanie tylko ta lista poleceń, która znajduje się pod pierwszym zadowalającym dopasowaniem napisu do wzorca.

Na liście wzorców można umieścić również "*"(gwiazdkę), która będzie wzorcem pasującym do każdego napisu, jeżeli poprzedzające dopasowania nie powiodły się.

Należy zwrócić uwagę na:

) – (okrągły nawias zamykający) kończący każdy wzorzec,

;; – (podwójny średnik) kończący każdą listę poleceń,

esac – słowo kluczowe zamykające całą konstrukcję.

6.7.3. Instrukcja iteracyjna while

Jeżeli w skrypcie zachodzi potrzeba wielokrotnego powtórzenia jednej lub wielu instrukcji, można skorzystać z instrukcji iteracyjnych. Jako pierwsza zostanie omówiona instrukcja while.

Składnia:

```
while polecenie
do
polecenia
done
```

Podobnie do instrukcji if, polecenie może być instrukcją test albo dowolnym innym poleceniem ustawiającym status zakończenia. Pętla while będzie powtarzała listę poleceń polecenia tak długo, jak długo spełniony będzie warunek określany na podstawie wyniku działania polecenia polecenie.

Przykład:

```
#!/bin/bash
#skrypt wyświetla wszystkie argumenty
#nawet jeżeli jest ich więcej niż 9
while test -n "$1"
do
    echo -n "$1, "
    shift
done
echo
```

Jeżeli uruchomimy skrypt podając mu jako argument * (gwiazdkę), wówczas zostaną do niego przekazane nazwy wszystkich plików z bieżącego katalogu. Może być ich oczywiście więcej niż 9, więc aby uzyskać dostęp do nazwy dziesiątej i następnych można skorzystać z polecenia shift. Demonstrowany skrypt wyświetla wszystkie nazwy dodając za każdą z nich ", " (przecinek i spację). Znana już instrukcja echo w przypadku użycia opcji -n nie przechodzi do nowego wiersza.

Przykład pętli, której wykonywanie może przerwać tylko naciśnięcie klawiszy $\mathbf{Ctrl}{+}\mathbf{C}{:}$

```
while true
do
polecenia
done
```

W razie potrzeby całość można zapisać w jednym wierszu:

while true; do polecenie; done

(należy zwrócić uwagę na średniki).

6.7.4. Instrukcja iteracyjna until

Instrukcja until jest tak podobna do instrukcji while, że wystarczą dwa przykłady.

Tę pętlę przerwą wyłącznie klawisze Ctrl+C:

until false; do polecenie; done

A to druga wersja skryptu wyświetlającego wszystkie argumenty:

```
#!/bin/bash
#skrypt wyświetla wszystkie argumenty
#nawet jeżeli jest ich więcej niż 9
until test -z "$1"
do
    echo -n "$1, "
    shift
done
echo
```

Należy zauważyć, że w poleceniu test użyto opcji -z a nie -n, jak to miało miejsce w przypadku while.

6.7.5. Instrukcja iteracyjna for ... in

Składnia instrukcji jest następująca:

```
for zmienna in lista
do
polecenia
done
```

i oznacza ona, że zmienna przyjmie kolejno wszystkie wartości z listy.

Przykład:

```
for dzien in poniedzialek wtorek sroda czwartek piatek
sobota niedziela
do
    echo $dzien
```

done

wyświetli w kolejnych wierszach nazwy dni tygodnia.

Wszystkie pliki źródłowe programów w języku C można przetworzyć w następujący sposób:

```
for plik in *.c
do
    #polecenia przetwarzania pliku
done
```

Instrukcja **for** ma jeszcze jeden wariant składniowy, który pokazuje przykład:

```
#!/bin/bash
#skrypt wyświetla wszystkie argumenty
#nawet jeżeli jest ich więcej niż 9
for argument
do
    echo -n "$argument, "
done
echo
```

Pominięcie słowa kluczowego in i określenia listy spowoduje, że zmienna argument przyjmie kolejno wartości wszystkich argumentów wywołania skryptu.

Dokładnie ten sam efekt można osiągnąć, odwołując się do listy "\$@" przechowującej wszystkie argumenty wywołania skryptu:

```
for argument in "$@"
...
```

6.7.6. Instrukcja iteracyjna for

Interpreter poleceń **bash** ma również zaimplementowaną instrukcję **for** o składni zbliżonej do tej, znanej z języka C. Pokazuje to poniższy przykład:

```
#!/bin/bash
#skrypt wyświetla wszystkie argumenty wywołania
#nawet jeżeli jest ich więcej niż 9
#w postaci ponumerowanej listy
xx=$#
for((x=1; x<=xx; x++))
do
     echo $x "$1"
    shift
done</pre>
```

Wyzwaniem dla czytelnika niech będzie odpowiedź na pytanie dlaczego w przykładzie nie wystąpił zapis:

for((x=1; x<=\$#; x++))</pre>

6.8. Pliki inicjalizacyjne powłoki bash

Istnieje kilka plików odpowiedzialnych za inicjalizację interpretera poleceń (powłoki). Niektóre z nich (/etc/profile, /etc/bashrc) mogą być skonfigurowane dla konkretnej instalacji systemu operacyjnego. Inne

(~/.profile, ~/.bash_profile, ~/.bashrc) mogą być modyfikowane przez użytkownika i inicjalizować interpreter poleceń zgodnie z jego preferencjami.

Zgodnie z przyjętą regułą pliki /etc/profile i ~/.profile mogą być czytane również przez inne interpretery poleceń niż bash, stąd też nie powinny zawierać poleceń specyficznych dla tej powłoki, a jedynie podzbiór wspólny dla wszystkich powłok.

W przypadku dostępu związanego z logowaniem bash poszukuje plików konfiguracyjnych w następującej kolejności ~/.bash_profile, ~/.bash_login, ~/.profile, i ustawia konfigurację w oparciu o pierwszy wyszukany plik.

W przypadku uruchamiania okna konsoli (terminala) przez zalogowanego użytkownika (na przyklad z trybu graficznego), konfiguracja ustawiana jest w oparciu o plik ~/.bashrc. Obecnie właśnie taki dostęp do trybu konsolowego jest najbardziej powszechny i najczęściej modyfikowany jest właśnie ten plik.

Pliki inicjalizacyjne są skryptami powłoki. Mogą zawierać wszystkie dotychczas omówione polecenia, jednak zawierają z reguły wąski podzbiór poleceń, które powinny być wykonane jednokrotnie, ale każdorazowo przy uruchamianiu powłoki. Należy do nich zainicjowanie niektórych zmiennych środowiskowych (np. PATH, EDITOR, PAGER), zdefiniowanie sposobu zgłoszenia (tzw. *prompt* w zmiennych środowiskowych PS1 i PS2) i ustawienie tzw. aliasów.

Zmienna środowiskowa PATH przechowuje listę katalogów, w których powłoka **bash** wyszukuje pliki wykonywalne programów. Użytkownik może życzyć sobie, by na tej liście znalazły się katalogi istotnie dla niego, a nie mające znaczenia dla innych użytkowników. Odpowiedniej modyfikacji można dokonać wydając polecenie:

PATH=\$PATH:mojKatalog

Od tej chwili powłoka bash będzie uwzględniała również mojKatalog podczas poszukiwania programów. Aby taka zmiana odbywała się automatycznie, podczas każdego uruchamiania konsoli, należy instrukcję modyfikacji ścieżki zapisać w pliku ~/.bashrc.

Polecenie **alias** umożliwia zdefiniowanie napisu, który jeżeli pojawi się jako pierwszy w linii poleceń, zostanie zastąpiony przez inny napis, jeszcze zanim powłoka zacznie interpretować polecenie.

Prosty przykład:

Użytkownik najczęściej listuje zawartość katalogów poleceniem $\tt ls$ –la. Po wydaniu polecenia:

alias la="ls -la"

zamiast 5 znaków $\tt ls -la,$ będzie mógł wpisywać tylko 2 znaki $\tt la$ i osiągnie ten sam efekt.

Wpisując powyższe polecenie do pliku /.bashrc użytkownik uwolni się od każdorazowego definiowania aliasu.

Zdefiniowany alias la można usunąć wydając polecenie:

unalias la

6.9. Podsumowanie

Przedstawiony w niniejszym rozdziale zakres wiedzy jest więcej niż wystarczający, by swobodnie rozpocząć programowanie skryptów powłoki bash.

Rozdział 7

Połączenia zdalne

| 7.1. | Dostęp do odległego komputera | 120 |
|------|---------------------------------------|-----|
| 7.2. | Przesyłanie plików między komputerami | 122 |
| 7.3. | Połączenia z systemu Windows | 125 |

7.1. Dostęp do odległego komputera

Historycznie najstarszym (rok 1969) protokołem umożliwiającym połączenie z odległą maszyną po sieci TCP jest **telnet**. Protokół opracowano w czasach, gdy sieci komputerowe miały charakter lokalny, w zasadzie zamknięty i względy bezpieczeństwa nie miały aż tak wielkiego znaczenia. Cała transmisja odbywa się bez żadnych zabezpieczeń i może być podpatrzona i przechwycona. Z tego względu obecnie protokół ten znajduje głównie niszowe zastosowania (na przykład do lokalnej konfiguracji osprzętu sieciowego).

Uruchamiając telnet można zażądać połączenia z konkretnym portem odległej maszyny. Cecha ta umożliwia wykorzystanie programu telnet do testowania protokołów sieciowych.

Jak widać mimo wszystko jest celowe posiadanie elementarnego zasobu wiedzy, który umożliwi korzystanie z tego programu.

Aby możliwe było korzystanie z protokołu telnet, wymagany jest program serwera na odległej maszynie i program klienta na maszynie łączącej się z serwerem.

W systemie GNU/Linux klient protokołu telnet, jak tego można oczekiwać, nosi również nazwę telnet. Program można uruchomić bez żadnych argumentów, wówczas uzyskujemy zgłoszenie w postaci:

telnet>

Program sygnalizuje w ten sposób gotowość do przyjmowania poleceń. Listę dostępnych poleceń wyświetli polecenie ? (albo help).

Polecenia mogą być skracane, o ile nie powoduje to niejednoznaczności. Więcej informacji na temat dowolnego polecenia można otrzymać pisząc ? polecenie (help polecenie).

Najważniejszym poleceniem jest **open**, wymagające podania nazwy albo adresu IP serwera. Jak wspomniano wyżej możliwe jest również podanie portu TCP, z którym **telnet** ma się łączyć (standardowo jest to port 23). Połączenie z serwerem zestawiane jest z przeprowadzeniem procedury autoryzacji (należy podać nazwę użytkownika i hasło). Po zestawieniu połączenia, **telnet** z punktu widzenia użytkownika staje się *przezroczysty* – po prostu możliwa jest praca na odległym systemie z wykorzystaniem jego powłoki (**bash**). Połączenie zakończy polecenie **exit**, **logout** albo naciśnięcie klawiszy Ctrl+D.

Program telnet można uruchomić w sposób bezzwłocznie zestawiający połączenie z serwerem (można w ten sposób pominąć polecenie open), podając nazwę albo adres IP serwera:

telnet serwer [port]

Można również podać od razu nazwę użytkownika:

telnet -l user serwer [port]

omawianego telnetu brak mechanizmów Główna wadą jest Obecnie bezpieczeństwa. powszechnie stosowanym protokołem zapewniającym szyfrowane, bezpieczne połączenie jest ssh – (secure shell). W systemie GNU/Linux zaimplementowana jest wersja OpenSSH tego protokołu, a jego klient jak łatwo się domyśleć nazywa się **ssh**. Program można uruchomić jednym z poleceń:

```
ssh serwer
ssh -l user serwer
ssh user@serwer
```

Każdy wariant dopuszcza użycie opcji -p port dla wskazania innego niż domyślny (22) numeru portu, na którym serwer oczekuje na połączenia.

Do zestawienia połączenia jest potrzebne podanie poprawnych nazw użytkownika i jego hasła.

Jeżeli połączenie z odległą maszyną jest zestawiane pierwszy raz, program upewni się, czy rzeczywiście takie były intencje (na zadane pytanie należy odpowiedzieć pełnym napisem **yes**) i zapisze informacje identyfikujące odległy komputer (*RSA key fingerprint*) w pliku ~/.ssh/known_hosts. W ten sposób przy kolejnych połączeniach możliwe będzie wykrycie próby podszywania się pod odległy system (na przykład przez zafałszowanie wpisów w serwerach DNS).

Po zestawieniu połączenia program **ssh** staje się całkowicie "przezroczysty" i pracując na odległym systemie nie odczuwamy różnicy w stosunku do pracy na systemie lokalnym.

Program **ssh** jest jednym z podstawowych narzędzi umożliwiających studentom pracę na udostępnionych im serwerach.

Oprócz omówionych programów telnet i ssh do zestawienia połączenia z odległą maszyną można próbować wykorzystać jeszcze programy **rsh** (*remote shell*) i **rlogin** (*remote login*). Składnia prezentowana na poniższych przykładach:

rsh -l user serwer rlogin -l user serwer

jest uniwersalna dla tego typu programów.

Dzisiaj, w dobie szerokopasmowego internetu na znaczeniu zyskują programy klasy **vnc** (*virtual network computing*). Przykładem zastosowania tej idei jest **Zdalny pulpit**, który oferując dostęp do odległych komputerów w trybie graficznym daje złudzenie pracy lokalnej na odległym systemie.

W niniejszym skrypcie programy tego typu nie będą omawiane. Zagadnienie to powinno stać się jednak przedmiotem samodzielnej dociekliwości czytelników.

7.2. Przesyłanie plików między komputerami

Najprostszym co do mechanizmu działania programem umożliwiającym kopiowanie plików pomiędzy komputerami jest program **rcp** (*remote copy*). Ponieważ program nie zapewnia zadowalających mechanizmów bezpieczeństwa, trudno znaleźć systemy komputerowe, w których obecnie można ten program rzeczywiście zastosować (poza lokalnym kopiowaniem plików). Z tego powodu program nie będzie omawiany.

Programem, który w całości realizuje funkcje programu rcp w sposób bezpieczny (korzystając z protokołu ssh) jest program scp (*secure copy*). Podstawowa forma wywołania:

scp user_1@komputer_1:plik_1 user_2@komputer_2:plik_2

Działanie programu przypomina omawiany wcześniej program cp, jedynie nazwa pliku może być poprzedzona identyfikatorem użytkownika (user_1, user_2) i nazwą systemu komputerowego (komputer_1, komputer_2), zgodnie z pokazaną składnią (należy zwrócić uwagę na znaki C i :). Zależnie od kierunku kopiowania plik_1 albo plik_2 jest plikiem lokalnym. Dla tego pliku fragment user_?Ckomputer_?: jest pomijany (program odróżnia pliki lokalne od zdalnych po obecności znaku : (dwukropka)). W trakcie nawiązywania połączenia program pyta o hasło użytkownika, stąd też nie jest możliwe przeprowadzenie nieuprawnionego kopiowania.

Oprócz funkcji kopiowania plików programu cp, scp przejął również niektóre jego opcje (na przykład -p lub -r). Dodatkową użyteczną opcją może okazać się -1 – ograniczenie wykorzystywanego pasma (podawane w kilobitach na sekundę).

ftp (file transfer protocol) – program stanowiący interfejs dla protokołu transferu plików ftp. Umożliwia kopiowanie plików pomiędzy lokalnym, a zdalnym systemem plików. Na komputerze zdalnym musi być uruchomiony program – tak zwany serwer protokołu ftp. Program klienta protokołu ftp można uruchomić jednym z poleceń:

ftp ftp serwer ftp user@serwer gdzie **user** jest nazwą użytkownika, a **serwer** jest nazwą albo adresem IP zdalnego systemu komputerowego. Do nawiązania połączenia potrzebna jest znajomość nazwy użytkownika i poprawnego hasła.

Po uruchomieniu zgłoszeniem:

ftp>

program sygnalizuje gotowość jego interpretera poleceń. Każde polecenie może być skrócone, o ile tylko nie spowoduje to niejednoznaczności. Klawisz **Tab** ma przypisaną funkcję autouzupełnienia.

Program ma własny system pomocy. Listę wszystkich dostępnych poleceń wyświetli polecenie ? albo help. Pomoc dotyczącą konkretnego polecenia wyświetli ? polecenie (albo help polecenie). Składnię konkretnego polecenia wyświetli polecenie usage polecenie.

Najważniejsze polecenia (niektóre dublują się):

- ! (wykrzyknik) wywołuje powłokę bash, aby wrócić do programu ftp należy zakończyć działanie powłoki poleceniem exit albo klawiszami Ctrl+D,
- ! polecenie wywołuje polecenie systemu operacyjnego, po zakończeniu wykonywania polecenia wraca do programu ftp,
- open, ftp nawiązuje połączenie z serwerem,
- close, disconnect zamyka sesję z serwerem,
- bye, exit, quit (również klawisze Ctrl+D) zamyka sesję z serwerem i kończy pracę programu,
- rate ogranicza wielkość wykorzystywanego przez program pasma (w bajtach na sekundę),
- pwd wyświetla nazwę bieżącego katalogu na serwerze,
- lpwd wyświetla nazwę bieżącego katalogu lokalnego,
- mkdir zakłada nowy katalog na serwerze,
- rm, rmdir usuwa katalog na serwerze,
- cd zmienia bieżący katalog na serwerze,
- lcd zmienia bieżący katalog na maszynie lokalnej,
- dir, 1s listuje pliki na *serwerze* (wyświetla zawartość katalogu),
- put, send wysyła plik na serwer,
- mput pliki wysyła na serwer pliki zadane listą i/lub maską,
- get, recv pobiera plik z serwera,
- mget pliki pobiera z serwera pliki zadane listą i/lub maską,
- newer pobiera plik z *serwera* pod warunkiem, że jest on młodszy od pliku lokalnego,
- delete usuwa plik na serwerze,
- mdelete pliki usuwa na serwerze pliki zadane listą i/lub maską,
- less, more, page umożliwiają podejrzenie zawartości plików tekstowych na serwerze,

— rename – zmienia nazwę pliku na serwerze,

— ascii, binary – ustawia odpowiedni tryb transferu plików.

Pliki przesyłane protokołem ftp mogą być traktowane jako binarne albo tekstowe (ASCII). Kilka uwag na ten temat:

- program ftp rozpoznaje wiele formatów binarnych i tekstowych, i automatycznie ustawia odpowiedni dla nich tryb przesyłania,
- pomiędzy systemem zdalnym, a lokalnym może wystąpić różnica w sposobie kodowania plików tekstowych, zwłaszcza jeżeli chodzi o znaki zmiany wiersza. Program podczas przesyłania plików tekstowych może podjąć próbę odpowiedniej konwersji kodowania,
- siłowa próba transferu plików binarnych w trybie *ascii* z pewnością utworzy uszkodzoną (zazwyczaj w sposób nieodwracalny) kopię.

Protokół ftp jest protokołem starym (starszym niż internet). Cała transmisja (łącznie z przekazaniem identyfikatora użytkownika i jego hasła) odbywa się bez żadnych zabezpieczeń. Możliwe jest więc podejrzenie i przechwycenie przesyłanych danych. Wady tej nie ma program sftp (*secure ftp*), który naśladuje zachowanie programu ftp, jednak zestawia bezpieczne połączenie szyfrowane protokołem ssh. Program sftp można uruchomić jednym z poleceń:

sftp serwer sftp user@serwer

Nie ma polecenia **open** znanego z **ftp**. W pierwszym przypadku program użyje nazwy użytkownika **user** z lokalnego systemu przechowywanego w zmiennej środowiskowej **\$USER**.

W przypadku pierwszego połączenia z odległą maszyną, obowiązują uwagi poczynione przy omawianiu programu **ssh** co do weryfikacji klucza RSA.

System pomocy programu sftp rozwiązany jest inaczej niż w programie ftp. Dostępne jest co prawda polecenie ? (albo help) wyświetlające listę dostępnych poleceń łącznie ze składnią, ale nie funkcjonuje wariant: ? polecenie (help polecenie). W przypadku niewłaściwego użycia któregokolwiek polecenia, zostanie wyświetlona instrukcja użycia właściwego.

Zestaw ważniejszych poleceń:

- ! (wykrzyknik) wywołuje interpreter poleceń bash, aby wrócić do programu sftp należy zakończyć jego działanie (exit, Ctrl+D),
- ! polecenie wywołuje polecenie systemu operacyjnego, po zakończeniu wykonywania polecenia wraca do programu sftp,
- bye, exit, quit (również klawisze Ctrl+D) kończy pracę programu,

- pwd, lpwd wyświetla nazwę bieżącego katalogu (odpowiednio na *serwerze* albo na *maszynie lokalnej* – podobnie w przypadku następnych trzech poleceń),
- cd, lcd zmienia bieżący katalog,
- mkdir, lmkdir zakłada nowy katalog,
- 1s, 11s listuje zawartość katalogu,
- rm plik usuwa plik na serwerze,
- rmdir katalog usuwa katalog na serwerze,
- rename stara_nazwa nowa_nazwa zmienia nazwę pliku na *serwerze*.
- Oczywiście najważniejsze są polecenia odpowiedzialne za transfer plików:
- get [-P] plik_serwera [plik_lokalny] pobiera plik_serwera [ewentualnie nadając mu nazwę plik_lokalny],
- put [-P] plik_lokalny [plik_serwera] wysyła plik_lokalny [ewentualnie nadając mu nazwę plik_serwera].

W dwóch ostatnich poleceniach plik_serwera i plik_lokalny mogą być maskami nazw plików, a użycie opcji -P spowoduje (o ile będzie to możliwe) skopiowanie również atrybutów plików (uprawnienia, czasy modyfikacji i dostępu).

wget (*web get*) – program służący do pobierania plików z internetu z wykorzystaniem protokołów http i ftp. Podstawowa forma uruchomienia:

wget [opcje] [adres]

gdzie adres lokalizuje plik zgodnie ze standardem URL. W przypadku uruchomienia, bez żadnych opcji wyniki zwrócone przez serwer zostaną zapisane w pliku, którego nazwa zostanie utworzona na podstawie adresu, a komunikaty programu są wyprowadzane na standardowe wyjście.

Bardziej użyteczne opcje:

- -o plik_komunikatow komunikaty programu zastaną zapisane w pliku o zadanej nazwie,
- -O plik_wyjsciowy dane zwrócone przez serwer zostaną zapisane w pliku o zadanej nazwie.

Użycie odpowiednich opcji umożliwia dostęp do serwerów wymagających *autoryzacji* (użytkownik i hasło), obsługę formularzy i tzw. *cookies* oraz wiele, wiele innych.

7.3. Połączenia z systemu Windows

Standardowy system Windows daje wyjątkowo skromne możliwości połączenia z odległymi komputerami. W trybie konsolowym są dostępne

programy telnet i ftp, które można traktować jak okrojone wersje programów o tych samych nazwach znanych z GNU/Linuksa.

Program **telnet** można uruchomić z wiersza poleceń albo z okienka dialogowego **Uruchom** wydając polecenie **telnet**.

Wykaz dostępnych poleceń wyświetli nam polecenie ? albo help. Polecenia mogą być skracane, o ile nie spowoduje to niejednoznaczności. Najważniejsze polecenia to:

— quit – zakończ pracę programu telnet,

open serwer [port] – nawiązuje połączenie z serwerem [na porcie].
 telnet można uruchomić żądając bezzwłocznego nawiązania połączenia:

```
— telnet serwer [port]
```

— telnet -l user serwer [port]

Dla pełniejszego obrazu można zapoznać się z wcześniejszym opisem programu telnet dla systemu GNU/Linux.

W trybie graficznym do systemu Windows (do XP włącznie), w menu Akcesoria | Narzędzia Systemowe | Komunikacja można było znaleźć program Hyperterminal, który mógł pełnić rolę klienta protokołu telnet. Począwszy od Windows Vista program ten nie występuje w systemie Windows.

W trybie konsolowym systemu Windows do przesyłania plików można wykorzystać program **ftp**. Sposób uruchomienia konsolowego klienta protokołu **ftp** systemu Windows przewiduje dwa warianty:

ftp

ftp serwer

przy czym serwer może być nazwą albo adresem IP serwera.

Po uruchomieniu programu są dostępne następujące polecenia:

- --- ? (pytajnik), help -- wyświetla listę dostępnych poleceń,
- ? polecenie, help polecenie wyświetla pomoc dotyczącą konkretnego polecenia,
- ! (wykrzyknik) wywołuje interpreter poleceń cmd, aby wrócić do programu ftp należy zakończyć działanie cmd poleceniem exit,
- ! polecenie wywołuje polecenie systemu operacyjnego, po zakończeniu wykonywania polecenia wraca do programu ftp,
- open nawiązuje połączenie z serwerem,
- close, disconnect zamyka sesję z serwerem,
- bye, quit zamyka sesję z serwerem i kończy pracę programu,
- pwd wyświetla nazwę bieżącego katalogu na serwerze,
- mkdir zakłada nowy katalog na serwerze,
- rm, rmdir usuwa katalog na serwerze,
- cd zmienia bieżący katalog na serwerze,

- lcd zmienia bieżący katalog na maszynie lokalnej,
- dir, 1s listuje pliki na serwerze (wyświetla zawartość katalogu),
- put, send wysyła jeden plik na serwer,
- mput pliki wysyła na serwer pliki zadane listą i/lub maską,
- get, recv pobiera jeden plik z serwera,
- mget pliki pobiera z serwera pliki zadane listą i/lub maską,
- delete usuwa plik na serwerze,
- mdelete pliki usuwa na serwerze pliki zadane listą i/lub maską,
- rename zmienia nazwę pliku na serwerze,
- ascii, binary ustawia odpowiedni tryb transferu plików.

Uwagi dotyczące trybu transferu plików zostały zamieszczone przy opisie linuksowej wersji programu ftp.

Standardowo system Windows nie udostępnia żadnych narzędzi umożliwiających emulację terminala albo bezpieczny transfer plików w trybie okienkowym. Na szczęście na zasadach freeware są dostępne programy **PuTTY** i **WinSCP**. Pierwszy może pełnić funkcję terminala pracującego jako klient protokołu SSH, drugi umożliwia przesyłanie plików pomiędzy komputerami w protokołach SCP, SFTP i FTP.

Logowanie do zdalnego systemu z wykorzystaniem programu PuTTY wymaga podania dokładnie takich samych danych, jak w przypadku logowania programem **ssh** (rysunek 7.1).

| Real Putty Configuration | | × | | | | |
|--------------------------|--|---|--|--|--|--|
| Category: | | | | | | |
| 🖃 Session | Basic options for your PuTTY session | | | | | |
| Logging | Specify the destination you want to connect to | | | | | |
| Kevboard | Host <u>N</u> ame (or IP address) <u>P</u> ort | . | | | | |
| Bell | user@serwer 22 | | | | | |
| - Features | Connection type: | | | | | |
| 🖹 Window | <u> </u> | | | | | |

Rysunek 7.1: Logowanie programem PuTTY

W przypadku programu PuTTY można napotkać problemy z wyświetlaniem polskich znaków. Aby naprawić tę niedogodność należy sprawdzić poleceniem locale jaka strona kodowa ustawiona jest po stronie serwera, następnie kliknąć w pasek tytułowy programu PuTTY prawym klawiszem myszy i wzorując się na rysunku 7.2 odpowiednio zmienić ustawienia.

Logowanie w przypadku wykorzystania programu WinSCP przebiega bardzo podobnie (rysunek 7.3).



Rysunek 7.2: Ustawienie kodowania w programie PuTTY

| WinSCP Login | | <u>? ×</u> |
|--|---|---|
| Session Stored sessions Environment Directories SSH Preferences | Session Host name serwer User name P user Private key file | Po <u>r</u> t number 22 = assword |
| | Protocol <u>File protocol</u> SFTP | ▼ IZ Allow SCP fallback |

Rysunek 7.3: Logowanie programem WinSCP

W programie WinSCP można łatwo rozpoznać interfejs wzorowany na programie Norton Commander, a tym samym przypominający program Midnight Commander (rysunek 7.4).

| ⁵ a | | - WinSCP | | | | | | | _10 | ' |
|--|-----------------------|----------------|------------------------------|-------|------------------------|---------|-----------|------------|----------|-----|
| <u>Remote</u> Mark E | iles <u>⊂</u> ommands | Session Option | s <u>L</u> ocal <u>H</u> elj | c | | | | | | |
| • 🛛 🖓 • 🛛 | A 🔹 📀 🔤 | 🖉 😤 王 | - ∀ ¢ | ØØ | Default 🔹 🌀 | - | | | | |
| ISI 🖂 | | - 6 C A | 🔊 🚗 😘 | | D: D. NTES | | | i 🖪 🖻 🚗 | 20 | |
| /1181 | | | | 1 | D:\Java\idk1.6.007\bin | | | | 40 | |
| Name Ext | Size | Changed | Rights | Owner | Name Ext | Size | Type | Changed - | | |
| a | | 2010-08-19 | rwxr-xr-x | root | 1 | | Parent | 2008-10-28 | 09:18:06 | |
| Co.snan | | 2007-09-14 | rwxrwxr-x | root | Smsvcr71.dl | 348 160 | Rozsze | 2008-06-10 | 05:44:26 | |
| bin | | 2008-07-04 | rwxr-xr-x | root | S beanreg.dl | 29 184 | Rozsze | 2008-06-10 | 05:27:15 | |
| Compat | | 2006-10-10 | rwxr-xr-x | root | Ni.dl | 77 824 | Rozsze | 2008-06-10 | 05:27:15 | |
| ames | | 2008-07-04 | rwxr-xr-x | root | HtmlConverter.exe | 45 056 | Aplikacja | 2008-06-10 | 03:32:37 | |
| home | | 2010-04-26 | rwxr-xr-x | root | 🕌 javaws.exe | 139 264 | Aplikacja | 2008-06-10 | 03:32:37 | |
| Cinclude | | 2008-07-04 | rwxr-xr-x | root | 🖉 packager.exe | 73 728 | Aplikacja | 2008-06-10 | 03:32:37 | |
| ib 🗀 | | 2008-07-04 | rwxr-xr-x | root | extcheck.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | - L |
| Dibdata | | 2005-08-10 | rwxr-xr-x | root | idj.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| ibexec 🗀 | | 2008-07-04 | rwxr-xr-x | root | jar.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | _ |
| Colocal | | 2008-07-04 | rwxr-xr-x | root | jarsigner.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| Cobj | | 2005-08-10 | rwxr-xr-x | root | 🛃 java.exe | 135 168 | Aplikacja | 2008-06-10 | 03:10:58 | |
| (Comports | | 2010-09-06 | rwxr-xr-x | root | javac.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| Corts_old | | 2007-05-13 | rwxr-xr-x | root | javadoc.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| 🚞 sbin | | 2008-07-04 | rwxr-xr-x | root | 🗂 javah.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| 🚞 share | | 2005-08-10 | rwxr-xr-x | root | javap.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| C src | | 2008-04-18 | rwxr-xr-x | root | java-rmi.exe | 25 088 | Aplikacja | 2008-06-10 | 03:10:58 | |
| 🚞 tmp | | 2009-10-14 | rwxr-xr-x | root | 🕌 javaw.exe | 135 168 | Aplikacja | 2008-06-10 | 03:10:58 | |
| 🔂 X11R6 | | 2007-06-06 | rwxr-xr-x | root | jconsole.exe | 26 112 | Aplikacja | 2008-06-10 | 03:10:58 | |
| X11R6_OLD | | 2005-08-16 | rwxr-xr-x | root | jdb.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| | | | | | jhat.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | |
| | | | | | jinfo.exe | 25 600 | Aplikacja | 2008-06-10 | 03:10:58 | - |
| • | | | | • | 1 ग | | | | 1 | • 🗐 |
| 0 B of 0 B in 0 of 19 208 KB of 2 091 KB in 2 of 49 | | | | | | | | | | |
| 💷 F2 Rename 📝 F4 Edit 📫 F5 Copy 🗳 F6 Move 🍲 F7 Create Directory 🗡 F8 Delete 😁 F9 Properties 🧵 F10 Quit | | | | | | | | | | |
| | | | | | | | A s | FTP-3 🗐 | 0:06:20 | 6 / |

Rysunek 7.4: Interfejs programu WinSCP
Rozdział 8

EDYTOR EMACS

| 8.1. | Wprowadzenie | 132 |
|------|--|------------|
| 8.2. | Instalacja i konstruowanie poleceń edytora | 133 |
| 8.3. | Podstawy edycji | 136 |
| 8.4. | Dostosowywanie edytora | 137 |
| 8.5. | Podsumowanie | 139 |
| 8.6. | Zadania | 139 |

8.1. Wprowadzenie

Podczas pracy z systemem operacyjnym Linux prędzej czy później zaistnieje konieczność edycji plików tekstowych. Edytor tekstowy będzie potrzebny zarówno do obróbki plików konfiguracyjnych jak też do pracy z kodem źródłowym. W edytorze tekstowym można tworzyć również wszystkie inne, "zwyczajne" teksty, takie jak książki, artykuły naukowe, listy i im podobne. Przekonamy się o tym przy okazji omawiania systemu składu IATEX. Wśród wielu niewiadomych pojawiających się u nowych użytkowników systemu, jedno jest pewne: trzeba mieć dobry edytor.

Emacs jest prawie tak stary jak Unix. Historia edytora sięga przełomu lat siedemdziesiątych i osiemdziesiątych ubiegłego stulecia, kiedy to Ken Thompson wraz ze współpracownikami ulepszali kolejne wersje edytorów obecnych w ówczesnych wersjach systemu. W 1976 roku powstał między innymi używany do dziś edytor vi (skrót od angielskiego visual editor) i kilka innych mniej lub bardziej zaawansowanych edytorów, skupionych wokół środowisk wywodzących się z laboratoriów Bella i Uniwersytetu Berkeley w Kalifornii. W tym samym czasie, na wschodzie Stanów Zjednoczonych, w Laboratorium Sztucznej Inteligencji MIT Richard Stallman zainspirowany możliwościami nowych edytorów wraz ze współpracownikami doprowadził do stworzenia najpotężniejszego edytora w historii o nazwie Emacs (od angielskiego Editing MACroS).

Użycie przymiotnika "najpotężniejszy" nie stanowi przesady. Do dziś profesjonalni informatycy tworzący oprogramowanie bardzo często korzystają z możliwości Emacsa. Edytor Emacs może być wykorzystywany zarówno do tworzenia kodu źródłowego oprogramowania, jak też do pisania powieści. W edytorze Emacs da się zrobić praktycznie wszystko, o czym tylko można przy okazji edycji zaawansowanych dokumentów tekstowych pomyśleć. Nie ma na świecie edytora, który w tak elastyczny sposób mógłby być spersonalizowany przez poszczególnych użytkowników. W celu konfiguracji wykorzystuje się polecenia języka LISP stworzonego na potrzeby sztucznej inteligencji. Emacs posiada wszystkie przymioty właściwe dla "prostszych" edytorów zarówno tekstowych jak i graficznych, mamy więc m.in. moduły sprawdzania pisowni, wyróżniania składni itp. Jednak na tym nie koniec: Edytor Emacs to taki edytor, z którym można zagrać w szachy, sprawdzić pocztę elektroniczną, a w razie problemów porozmawiać z sztucznointeligentnym psychoterapeutą.

Niektórzy żartobliwie podtrzymują, że Emacs to właściwie system operacyjny, na który nakładką jest Unix. W środowisku powstał nawet kościół Emacsa. Członkiem kościoła zostaje każdy, kto po trzykroć powtórzy słowa: "There is no system but GNU, and Linux is one of its kernels" (Nie ma systemu prócz GNU, a Linux jest jednym z jego jąder). Patronem kościoła jest Święty IGNUcy (Rys. 8.1).



Rysunek 8.1: Richard Stallman jako Święty IGNUcy

8.2. Instalacja i konstruowanie poleceń edytora

Z nieznanych przyczyn Emacs nie jest domyślnie dostarczany z dystrybucją Ubuntu.

W celu zainstalowania edytora i niekoniecznych, aczkolwiek przyjemnych dodatków (motywy kolorystyczne itp.) należy wykonać następujące polecenia:

sudo aptitude install emacs

oraz

sudo aptitude install emacs-goodies-el emacs-goodies-extra-el

Emacsa uruchamiamy w trybie graficznym (Rys. 8.2) wykonując polecenie w terminalu:

emacs

oraz w trybie tekstowym 8.3 - opcja niezmiernie użyteczna w przypadku łączenia się z komputerami zdalnymi - wykonując:

emacs -nw



Rysunek 8.2: Edytor Emacs uruchomiony w trybie graficznym

| Plik Edycja Widok Terminal Pomoc | | | | | |
|---|---|--|--|--|--|
| File Edit Options Buffers Tools Help | | | | | |
| Welcome to GNU Emacs, one component of the GNU/Linux operating system. | | | | | |
| _ | | | | | |
| Get help C-h (Hold down CTRL and press h) | | | | | |
| <u>Emacs manual</u> C-h r <u>Browse manuals</u> C-h i | | | | | |
| <u>Emacs tutorial</u> C-h t Undo changes C-x u | | | | | |
| Buy manuals C-h C-m Exit Emacs C-x C-c | | | | | |
| <u>Activate menubar</u> M-` | | | | | |
| (`C-' means use the CTRL key. `M-' means use the Meta (or Alt) key. | | | | | |
| If you have no Meta key, you may instead type ESC followed by the character.) | | | | | |
| Useful tasks: | | | | | |
| <u>Visit New File</u> <u>Open Home Directory</u> | | | | | |
| <u>Customize Startup</u> <u>Open *scratch* buffer</u> | | | | | |
| | | | | | |
| GNU Emacs 23.1.1 (i486-pc-linux-gnu, GTK+ Version 2.20.0) | | | | | |
| of 2010-03-29 on rothera, modified by Debian | | | | | |
| Copyright (C) 2009 Free Software Foundation, Inc. | | | | | |
| | | | | | |
| GNU Emacs comes with ABSOLUTELY NO WARRANTY; Type C-n C-W Tor <u>Tull details</u> . | | | | | |
| Emacs is Free SoftwareFree as in Freedomso you can redistribute copies | | | | | |
| b) Emacs and modily it; type c-n c-c to see <u>the conditions</u> . | | | | | |
| ype c-n c-o for information on <u>getting the tatest version</u> . | | | | | |
| For information about GNU Emace and the GNU system, type C h C a | | | | | |
| For information about GNU Emacs and the GNU system, type C-n C-a. | V | | | | |

Rysunek 8.3: Edytor Emacs uruchomiony w trybie tekstowym

Używanie Emacsa to nie tylko prestiż, ale i moc. Czytelnik po niedługim czasie korzystania z edytora przekona się o jego zaletach i praktycznie nieograniczonych możliwościach. Wtedy pierwszą czynnością po zainstalowaniu systemu operacyjnego będzie doinstalowanie ulubionego edytora.

Mimo, że edytor uruchomiony w trybie graficznym posiada menu, z poziomu którego da się wykonać większość najbardziej przydatnych poleceń, każdą komendę można też wydać używając odpowiednich skrótów klawiszowych. W skrótach klawiszowych tkwi potęga Emacsa, chociaż niektórzy uważają, że korzystanie z nich przypomina zachowania epileptycznego pająka [4]. Najważniejsze klawisze do generowania poleceń w edytorze Emacs to: *Ctrl, Esc, Alt* i *Shift.* Czasami skrót EMACS rozwija się właśnie od nazw tych klawiszy, gdzie dla litery M zarezerwowano meta-klawisz. (Istnieje jeszcze humorystyczne rozwinięcie tego skrótu: *Eight Megabytes And Constantly Swapping* oznaczające w wolnym tłumaczeniu konieczność korzystania partycji wymiany nawet w komputerach wyposażonych w 8 megabajtów pamięci operacyjnej! W dawnych czasach Emacs był rzeczywiście pamięciożerny, dziś więcej niż 8 MB pamięci posiada prawdopodobnie każda kuchenka mikrofalowa.

Należy zapamiętać, że *Ctrl* działa podobnie do *Shift* tylko w wtedy gdy jest wciśnięty i przytrzymany. Następnie naciskamy dodatkowy klawisz lub klawisze i kończymy całą komendę zwalniając *Ctrl*. Można to sprawdzić wychodząc z Emacsa przez wykonanie polecenia Ctrl+x+c. Uwaga: zarówno tu jak i w dalszej części tekstu znak + oznacza konieczność wciśnięcia klawisza stojącego za nim, bynajmniej nie plusa samego w sobie. W przeciwnym przypadku osiągnęlibyśmy stadium epileptycznej stonogi.

Klawisze Alt i Esc działają w ten sposób, że należy najpierw je wcisnąć, potem puścić i dopiero nacisnąć klawisze konieczne do wykonania komendy. Można to wypróbować przechodząc do linii poleceń Emacsa wykonawszy polecenie Esc+x a następnie wpisując prośbę o pomoc help.

Koniecznie należy wspomnieć o pewnej specyficznej nomenklaturze używanej w nazewnictwie klawiszy przez samego Emacsa. Klawisz Esc oznaczany jest tam jako meta-klawisz M, klawisz Ctrl jako C. Na domiar złego zamiast intuicyjnego plusa - Emacs używa znaku minus. Zatem komendy wyjścia i wywołania linii poleceń w systemie pomocy Emacsa wyglądałyby następująco: C-x-c i M-x. My jednak w celach psychologicznych, specjalnie dla przerażonych czytelników będziemy używać intuicyjnego oznaczania klawiszy.

Należy zauważyć, że w chwili obecnej czytelnik potrafi już wejść do i wyjść z edytora. Może to już uznać za swego rodzaju sukces.

8.3. Podstawy edycji

W środowisku Emacs wiele rzeczy można zrobić na wiele sposobów. My jednak przytoczymy te, które uważamy za najbardziej intuicyjne dla typowego, początkującego użytkownika edytora.

Plik w Emacsie można otworzyć wywołując w terminalu polecenie:

emacs nazwa-pliku

z nazwą pliku jako parametrem.

Jeżeli jednak chcemy otworzyć jakiś plik pracując już w edytorze, powinniśmy skorzystać kombinacji klawiszy Ctrl+x Ctrl+f, co oznacza (dla przypomnienia) trzymając wciśnięty klawisz Ctrl wciśnij x, zwolnij Ctrl, a następnie trzymając wciśnięty Ctrl wciśnij f i zwolnij Ctrl. Na dole ekranu pojawi się wtedy linia rozpoczynająca się od słów "*Find file:*" i ścieżka katalogów, w której obecnie się znajdujemy. Wyszukując plik do otwarcia musimy podać miejsce, w którym ten plik się znajduje. Tu mamy pewien komfort w postaci działającego klawisza tabulacji, który uzupełnia w sposób inteligentny kolejne człony ścieżki, w taki sam sposób jak konsola systemu Linux.

Po edytorze Emacs poruszamy się na wiele sposobów. W trybie graficznym możemy skorzystać z myszki, jej rolek i przycisków. Zarówno w trybie graficznym jak i w tekstowym powinny (może to zależeć od rodzaju powłoki) działać podstawowe klawisze do przemieszczania się w tekście, takie jak strzałki, *PageUp* oraz *PageDown*. Istnieje jednak pewna grupa bardziej zaawansowanych poleceń, które mogą być wykorzystywane w celu poruszania się w oknie edytora. Przytoczymy je z odpowiednimi skojarzeniami, które ułatwią zapamiętanie:

- Ctrl+a przejdź na początek bieżącego wiersza (bo a jest na początku alfabetu),
- *Ctrl+e* przejdź do końca następnego wiersza (e jak end),
- *Ctrl+f* przejdź do przodu o jeden znak (f jak forward),
- Ctrl+b przejdź do tyłu o jeden znak (b jak backward),
- *Ctrl+p* przejdź do poprzedniego wiersza (p jak previous),
- Ctrl+n przejdź do następnego wiersza (n jak next),
- *Ctrl+v* przewiń wyświetlaną zawartość o jeden ekran w dół,
- Esc+f przejdź do przodu o jedno słowo (skojarzenia? spójrz wyżej),
- $\mathit{Esc+b}$ przejdź do tyłu o jedno słowo,
- $\mathit{Esc+v}$ przewiń wyświetlaną zawartość o jeden ekran w górę,
- $\mathit{Esc+<}$ przejdź do początku bieżącego bufora,
- -- Esc+>- przejdź do końca bieżącego bufora.

Zaznaczanie tekstu realizujemy przy pomocy wciśniętego klawisza *Shift* i odpowiednich strzałek. Początek bloku, który zamierzamy zaznaczyć można też ustawić wykonując Ctrl+@ lub Ctrl+spacja, a następnie przenieść kursor do końca miejsca, które ma być zaznaczone. Cztery podstawowe polecenia edycji wykonujemy w sposób następujący:

— Esc+w - kopiuj,

— Ctrl+w - wytnij,

- Ctrl+y - wklej,

— $Ctrl+_{-} - cofnij.$

oraz polecenia zapisywania pliku:

- Ctrl+x+s zapisz,
- Ctrl+x+w zapisz jako.

Przeszukiwanie tekstu następuje po wykonaniu Ctrl+s. Na dole ekranu pojawia się wtedy linia rozpoczynająca od *I-Search:*. Przeszukiwanie rozpoczyna się w dół, wpisując kolejne litery ciągu o przeszukania edytor wyszukuje łańcuchy odpowiadające naszym kryteriom. Przeszukiwanie można przerwać wykonując Ctrl+g lub inne, dowolne polecenie Emacsa.

Podczas pracy w trybie tekstowym przydatne może okazać się polecenie wejścia do menu: Esc+'.

W razie kłopotów prawie z każdego miejsca da się wejść do trybu pomocy" Ctrl+x Ctrl+h.

8.4. Dostosowywanie edytora

Prędzej czy później zajdzie potrzeba personalizacji edytora. Czytelnicy obserwujący zaawansowanych użytkowników Emacsa powinni zauważyć, że praca każdego z nich może różnić się nieco. Okna edytorów mogą przybierać inne kolory niż standardowy, niektóre skróty klawiszowe mogą działać inaczej, system edycji może posiadać cechy jakie nie są obecne w domyślnych instalacjach. Plikiem konfiguracyjnym Emacs jest .emacs (UWAGA: nazwa pliku rozpoczyna się od kropki) znajdujący się w katalogu domowym użytkownika. Jeżeli plik o tej nazwie nie istnieje, należy go stworzyć. Właśnie w tym pliku można wpisać przy pomocy poleceń jezyka programowania LISP dowolne zmiany konfiguracji Emacs. I to dlatego, ze wzgledu na możliwości LISPA, Emacs jest najbardziej konfigurowalnym edytorem na świecie. Zanim pokażemy przykładową konfigurację Emacsa, nauczymy się zmieniać motyw kolorystyczny edytora. W tym celu należy wykonać polecenie: Esc+x przenoszace użytkownika do linii poleceń edytora. W linii tej (rozpoczynajacej sie od M-x) należy wpisać polecenie color-theme-select. Powinno otworzyć się okno, w którym użytkownik przy pomocy strzałek może wybrać sobie jeden z kilkudziesięciu motywów

kolorystycznych, najbardziej pasujący do jego preferencji. Po dokonaniu wyboru wychodzimy w okna selekcji motywu kolorystycznego zaznaczając opcję [*Quit*] na górze ekranu. Okaże się jednak, że po wyjściu z Emacsa, nasz ulubiony i wybrany motyw kolorystyczny nie zostanie zapamiętany. Co więcej, czasami ten sam motyw wygląda zupełnie inaczej w trybie graficznym niż w tekstowym. W pliku konfiguracyjnym **.emacs** możemy zdefiniować jaki motyw kolorystyczny ma być domyślnie uruchamiany w trybie graficznym, a jaki w trybie tekstowym. W tym celu wpisujemy doń dwa polecenia LISPa:

```
;; Ustawianie kolorow
(require 'color-theme)
(if window-system
(color-theme-subtle-hacker)
(color-theme-charcoal-black))
```

zauważając, że linie zawierające komentarz rozpoczynają się od podwójnego średnika. W gałęziach instrukcji warunkowej wpisujemy nazwy motywów według powyższego schematu.

Niezwykle pomocny jak zawsze jest moduł sprawdzania pisowni. Dlatego warto do pliku konfiguracyjnego Emacsa dodać następujące linie:

```
;; Sprawdzanie pisowni
(setq ispell-program-name "aspell")
(setq ispell-dictionary "polish")
(add-hook 'LaTeX-mode-hook 'flyspell-mode)
(add-hook 'LaTeX-mode-hook 'flyspell-buffer)
(defun turn-spell-checking-on ()
"Turn speck-mode or flyspell-mode on."
(flyspell-mode 1))
(add-hook 'text-mode-hook 'turn-spell-checking-on)
```

pamiętając o wcześniejszym do
instalowaniu silnika ${\bf aspell}$ i pakietów słownikowych:

```
sudo aptitude install aspell-pl
```

w systemie.

Sprawdzanie pisowni w całym pliku następuje po wykonaniu polecenia ispell z linii poleceń Emacsa wywoływanej przez Esc+x.

Przedstawiono tylko dwie możliwości dostosowywania edytora do własnych potrzeb. W Internecie można znaleźć wiele stron poświęconych temu zagadnieniu, chociażby [28].

8.5. Podsumowanie

Przedstawiono historię, podstawowe funkcje i część możliwości edytora. Stanowią one zaledwie niewielki ułamek tego, co Emacs potrafi. W miarę wzrostu poziomu zaawansowania użytkownicy będą samodzielnie odkrywać nowe funkcje i aspekty konfiguracji.

W historii informatki istnieje niewiele programów, które mogą poszczycić się trzydziestoletnim rozwojem. Emacs należy do tej elitarnej grupy, co więcej jest nierozerwalnie związany z ideami wolnego i otwartego oprogramowania.

Używając Emacsa nie tylko dotykamy pięknych kart historii, ale przede wszystkim korzystamy z narzędzia o potencjalnie największych możliwościach. Należy mieć świadomość, iż mimo rozwoju graficznych systemów edycji kodu, to w Emacsie wciąż powstaje wiele gier, oprogramowania i kolejnych wersji systemów operacyjnych. Widoczna na pierwszy rzut oka ascetyczna budowa i skomplikowany interfejs po pewnym czasie stają się zaletami, które trudno przecenić.

Osoby, które na poważnie chcą zająć się zaawansowanym użytkowaniem uniksowych systemów operacyjnych powinny zdecydować się na korzystanie z jednego, zawsze preferowanego edytora plików tekstowych i stopniowo osiągać w tym poziom mistrzostwa. Emacs wydaje się być tu najlepszym wyborem.

8.6. Zadania

Zadanie 1

Zainstaluj Emacsa z dodatkami w swoim systemie operacyjnym. Przećwicz wchodzenie i wychodzenie z edytora.

Zadanie 2

Stwórz plik emacs-testowy-1.txt. Wypełnij go dowolną, zawierającą kilka ekranów treścią. Przećwicz poruszanie się po tekście, kopiowanie, wycinanie i wklejanie fragmentów. Zapisz zmodyfikowany plik z nazwą emacs-testowy-2.txt.

Zadanie 3

Stwórz plik konfiguracyjny Emacsa i zmodyfikuj go w taki sposób, żeby przy uruchamianiu w trybie graficznym ustawiał się inny schemat kolorystyczny niż w trybie tekstowym.

Zadanie 4

Napisz program typu "Hello World!" w języku C++ wykorzystując Emacsa. Zanim zaczniesz pisać kod zapisz plik jako "hello.cc". Zauważysz jak pięknie Emacs koloruje składnię i ustawia wcięcia.

Zadanie 5

Wzbogać Emacsa o funkcję sprawdzania pisowni.

Zadanie 6

Postaraj się zawsze używać Emacsa jako domyślnego edytora plików tekstowych. Zostań mistrzem Emacsa!

Rozdział 9

Wykresy funkcji i wizualizacja danych w środowisku Gnuplot

| 9.1. | Wprowadzenie | 142 |
|------|---------------------------------|-----|
| 9.2. | Wykresy funkcji | 143 |
| 9.3. | Wizualizacja danych | 147 |
| 9.4. | Tworzenie skryptów dla Gnuplota | 151 |
| 9.5. | Podsumowanie | 152 |
| 9.6. | Zadania | 152 |

9.1. Wprowadzenie

Gnuplot jest doskonałym narzędziem wykorzystywanym do wizualizacji funkcji matematycznych oraz danych doświadczalnych. To elastyczne narzędzie występuje w wersjach dla wielu platform, od Linux przez OS/2, OSX, VMS po Windows. Gnuplot rozpowszechniany jest na zasadach licencji gwarantującej otwartość oprogramowania z możliwością zmian i modyfikacji kodu źródłowego w rozumieniu Free Software Foundation (FSF) i jako taki jest wolnym oprogramowaniem. Komunikacja z Gnuplotem odbywa się z wiersza poleceń lub w trybie wsadowym.

Rozwijane od ćwierć wieku środowisko (pierwszą wersję Gnuplota datuje się na 1986 rok) oferuje tworzenie dowolnych typów wykresów i wizualizacji zarówno dwu jak i trójwymiarowych. W środowisku specjalistów zajmujących się szeroko rozumianymi naukami ścisłymi panuje przekonanie, że jeśli "czegoś nie da się przedstawić przy pomocy Gnuplota to zwyczajnie jest to nie do przedstawienia". Krótko mówiąc Gnuplot to doskonały pomocnik zawsze i wszędzie tam, gdzie zachodzi potrzeba zrobienia profesjonalnych wykresów, w wektorowej jakości (istnieje oczywiście funkcja eksportowania produktów do grafiki rastrowej) oraz dzięki przetwarzaniu wsadowemu szybko i w praktycznie dowolnej liczbie. Biorąc pod uwagę elastyczność oprogramowania, możliwość przenoszenia skryptów między platformami, jakość generowanych wykresów oraz zerowy koszt można pokusić się o stwierdzenie, że Gnuplot jest najlepszym i najskuteczniejszym środowiskiem służącym statycznej wizualizacji danych.

W marcu 2010. ukazała się najnowsza stabilna wersja Gnuplota 4.4.0. Wprowadzono niewielkie zmiany w stosunku do przygotowanej dla Ubuntu 10.04 LTS wersji 4.2.

W standardowej kompilacji Ubuntu nie ma zainstalowanego Gnuplota. W celu instalacji wykonujemy polecenie:

sudo aptitude install gnuplot

Po zakończonej sukcesem instalacji Gnuplota uruchamiamy wykonując polecenie:

gnuplot

Jeśli wszystko poszło dobrze powinna uruchomić się konsola taka jak na Rys. 9.1.

Powrót z konsoli Gnuplota do konsoli Ubuntu następuje po wydaniu polecenia:

quit

Zatem potrafimy już zainstalować, a także wejść do i wyjść z Gnuplota.

```
Plik Edycja Widok Terminal Pomoc
gmwojcik@grace:~$ gnuplot
        G N U P L 0 T
        Version 4.2 patchlevel 6
        last modified Sep 2009
        System: Linux 2.6.32-23-generic
        Copyright (C) 1986 - 1993, 1998, 2004, 2007 - 2009
        Thomas Williams, Colin Kelley and many others
        Type `help` to access the on-line reference manual.
        The gnuplot FAQ is available from http://www.gnuplot.info/faq/
        Send bug reports and suggestions to <http://sourceforge.net/projects/gnu
plot>
Terminal type set to 'wxt'
gnuplot>
```

Rysunek 9.1: Konsola Gnuplota po pierwszym uruchomieniu

9.2. Wykresy funkcji

Teraz nauczymy się kreślić funkcje matematyczne zadane wzorem. Dwuwymiarowe wykresy funkcji tworzymy przy pomocy polecenia **plot**. Na przykład po wykonaniu polecenia:

plot x*sin(x)

otrzymujemy wykres funkcji $f(x) = x \sin(x)$ taki jak na Rys. 9.2.

W przypadku gdy chcemy nałożyć na siebie dwa lub więcej wykresów funkcji używamy operatora przecinka. Po wykonaniu polecenia:

```
plot x*sin(x), 2*x*cos(x)+1
```

otrzymamy wykresy funkcji xsin(x) oraz 2xcos(x) + 1 w tym samym układzie współrzędnych (Rys. 9.3).

W tym miejscu należy zaznaczyć, że w celu dokładniejszego przyjrzenia się lub wyeksponowania innego niż domyślny, wybranego fragmentu danego wykresu możemy zdefiniować odpowiednie ograniczenia dla osi odciętych oraz rzędnych układu współrzędnych. W tym celu wykorzystujemy nawiasy kwadratowe i dwukropek do podania odpowiednich zakresów. I tak na przykład polecenie:

```
plot [-2:7] [-6:12] x*sin(x), 2*x*cos(x)+1
```

spowoduje wyświetlenie fragmentu wykresu z Rys. 9.3 ograniczonego prostymix=-2, x=7, y=-6, y=12 (Rys. 9.4).



Rysunek 9.2: Wykres funkcji f(x) = xsin(x) wykonany przy pomocy polecenia plot



Rysunek 9.3: Wykresy funkcji xsin(x)ora
z2xcos(x)+1wykonane przy pomocy polecenia plot w tym samym układzie współrzędnych



Rysunek 9.4: Wykresy funkcji xsin(x)oraz2xcos(x)+1ograniczone prostymix=-2, x=7, y=-6, y=12

Trójwymiarowe wykresy funkcji generujemy wykorzystując polecenie **splot**. W ten sposób szybko możemy wykreślić funkcję $z = x^2 y$ (Rys. 9.5):

splot x*x*y

Na Rys. 9.2-9.5 w oczy rzuca się brak podpisów pod osiami. Tytuły osi współrzędnych można w łatwy sposób dodać w następujący sposób:

```
set xlabel "x"
set ylabel "y"
set zlabel "z"
```

odpowiednio dla osi $x,\,y$ iz.Na Rys. 9.6 przedstawiono wykres funkcji $z=e^{-x^2y^2}$ w przedziałach $x\in[-2,2],\,y\in[-2,2],\,z\in[0,1].$

Na wygenerowaną w sposób domyślny przez Gnuplota trójwymiarową siatkę można nałożyć kolory wygenerowane przez szybki algorytm **pm3d**. W tym celu wykonujemy polecenia:

set pm3d replot

gdzie dyrektywa **replot** nakazuje Gnuplotowi przerysowanie ostatniego wykresu z uwzględnieniem dodanych opcji (Rys. 9.7).



Rysunek 9.6: Trójwymiarowy wykres funkcji dwóch zmiennych z tytułami osi współrzędnych w określonych przedziałach x,y,z



Rysunek 9.7: Wykres funkcji $z=e^{-x^2y^2}$ z nałożonym motywem kolorów

Każdy wygenerowany przez Gnuplota wykres można wyeksportować do pliku postscriptowego. Należy wtedy wydać polecenia ustanawiające plik postscriptowy terminalem oraz definiujące jego fizyczną nazwę na dysku, a następnie wykreślić co potrzeba w znany już nam sposób:

```
set terminal postscript enhanced color
set output "wykres-testowy.eps"
set pm3d
splot [-3:3][-3:3][0:0.2] x*x*exp(-x*x)*y*y*exp(-y*y)
```

Czasami ze względów formalnych powinniśmy dostarczać grafikę monochromatyczną. Wtedy należałoby przy ustawianiu terminala postscriptowego opcję **color** zamienić na **mono**.

9.3. Wizualizacja danych

Nie zawsze funkcja matematyczna zdefiniowana jest zwykłym wzorem. Ze szkoły podstawowej powinniśmy pamiętać, że funkcję można przedstawić w postaci tabelki - zbioru argumentów i odpowiadających im wartości. Bardzo często wyniki danych eksperymentalnych przedstawiane są w postaci takiej tabelki, tyle tylko, że jej rolę pełni tu plik tekstowy zawierający nawet tysiące kolumn i wierszy. Wykorzystamy algoryt
m Padego aproksymujący funkcję $y = e^{-x}$ i zakodowany w języku C++.

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
ſ
float x,y,z1,z2,d;
d=0.1;
x=0.0;
for (int i=0;i<50;i++)
{
x += d;
y = exp(-x);
z1 = (6 - 2*x)/(6 + 4*x + x*x);
z^2 = (6 - 4xx + xx)/(6 + 2x);
cout << x << " " << y << " " << z1 << " " << z2 << endl;
}
return 0;
}
```

Kompilacji algorytmu i uruchomienia programu wraz ze strumieniowym przekierowaniem wyników do pliku **pade-output.dat** dokonujemy wykonując dwa polecenia w powłoce systemu:

g++ pade-approx.cc -o pade-approx ./pade-approx > pade-output.dat

Otrzymany plik pade-output.dat zawiera 50 wierszy o następującej strukturze:

0.1 0.904837 0.904836 0.904839 0.2 0.818731 0.818713 0.81875 0.3 0.740818 0.740741 0.740909 4.9 0.0074466 -0.0765974 0.65886 5 0.00673796 -0.0784313 0.687499

W ten sposób uzyskane z obliczeń numerycznych dane można wizualizować w Gnuplocie wykonując polecenie:

```
plot "pade-output.dat" using 1:2 with lines,\
> "pade-output.dat" using 1:3 with lines,\
> "pade-output.dat" using 1:4 with lines
```

gdzie parametry typu "**using 1:2 with lines**" oznaczają polecenie wykonania wykresu drugiej kolumny jako funkcji kolumny pierwszej (oraz trzeciej i czwartej względem pierwszej) i połączenie poszczególnych punktów linią. Otrzymany wykres ma postać 9.8. Parametr **lines** można zamienić na **points** (wykres zawierający same punkty), **boxes** (wykres słupkowy), **steps** (wykres schodkowy) oraz **impulses** (wykres impulsowy). Należy zwrócić szczególną uwagę na symbol *backslash*, który w przypadku długich linii poleceń służy do przełamywania ich zarówno w skryptach jak i w konsoli Gnuplota.



Rysunek 9.8: Wizualizacja pliku wyjściowego z aproksymacji Padego funkcji $y=e^{-x}.$ Wykreślono 2.,3. i 4. kolumnę pliku względem kolumny 1

Na zakończenie przedstawimy trójwymiarową wizualizację pliku zawierającego ściągawkę pt. tabliczka mnożenia. Taki plik można wygenerować przynajmniej na dwa sposoby: wpisać ręcznie (co może wydawać się morderczym zadaniem) albo napisać program w C++ i przekierować wynik jego działania na przykład do pliku **tabliczka.dat**. Główna funkcja programu ma następującą treść:

for (int i=1;i<=10;i++)
{</pre>

```
for (int j=1;j<=10;j++)
cout << i~<< "\t" << j << "\t" << i*j << endl;
cout << endl; // o~tu jest ta pusta linia
}</pre>
```

i należy zwrócić szczególną uwagę na "porcjowanie" danych w tym wypadku po $10~{\rm przez}$ wstawianie pustej linii co $10~{\rm wierszy}.$

Wygenerowany plik posiada strukturę:

i daje się wizualizować w sposób analogiczny do tworzenia funkcji dwóch zmiennych. Tylko trzeba pamiętać o pustych liniach porcjujących dane.

Teraz wykonując polecenia Gnuplota:

```
set pm3d
set xlabel "a"
set ylabel "b"
set zlabel "ab"
splot "tabliczka.dat" u 1:2:3 with lines
```

otrzymujemy trójwymiarową reprezentację tabliczki mnożenia z podpisanymi osiami i nałożonym kolorowaniem $\mathbf{pm3d}$ (Rys. 9.9).



Rysunek 9.9: Wizualizacja pliku tabliczka.
dat. Wykreślono zależność kolumny 3. od 1. i $\mathbf{2}$

9.4. Tworzenie skryptów dla Gnuplota

Wspomniano już, że Gnuplot umożliwia też przetwarzanie wsadowe. Często zdarza się, że chcemy wygenerować wiele wykresów wizualizujących na przykład wyniki uzyskane na drodze eksperymentu komputerowego. Załóżmy, że trzeba zrobić dajmy na to kilkaset wykresów wizualizujących pliki z danymi o podobnej strukturze. Wtedy byłoby wysoce nierozsądnym wpisywanie poszczególnych komend w linii poleceń, nie wspominając o czasie, który użytkownik musiałby poświęcić na wizualizację. Gnuplot jako potężne narzędzie i w tym miejscu wychodzi nam na przeciw. Okazuje się, że wystarczy wpisać ciąg poleceń w dowolnym pliku tekstowym (skrypcie), np. **skrypt.e**, a następnie uruchamiając Gnuplota z nazwą skryptu jako parametrem:

gnuplot skrypt.e

powodujemy sekwencyjne wykonanie wszystkich poleceń zawartych w tym pliku. Warto wspomnieć, iż do komentowania kodu w skryptach Gnuplota wykorzystywany jest symbol #.

9.5. Podsumowanie

W tym miejscu świadomy czytelnik zauważy, że skrypty Gnuplota w połączeniu z możliwościami skryptów powłoki systemu Linux stanowią środowisko, które jeśli chodzi o wizualizację danych doświadczalnych i szybkie generowanie doskonałej jakości wykresów nie ma sobie równych.

Gnuplot to setki instrukcji i tysiące możliwości. Bez wątpienia na temat Gnuplota można by napisać odrębny podręcznik. W Internecie istnieje wiele "samouczków" Gnuplota, które po zapoznaniu się przedstawionymi tu podstawami pozwolą na korzystanie z pakietu w sposób absolutnie zaawansowany. Jednym z najlepszych kursów Gnuplota z przykładami jest strona pana Kawano z Los Alamos National Laboratory - "Gnuplot - Not So Frequently Asked Questions" [29].

9.6. Zadania

Zadanie 1

Wykreśl funkcję y = 2sin(x)cos(x) + 7.

Zadanie 2

Wykreśl w jednym układzie współrzędnych funkcje: $y = -2x^2 + 5$, $y = x^2 - 7$ oraz y = sin(2x) dla $x \in [-5, 5]$.

Zadanie 3

Wykreśl funkcję $f(x,y) = x^2y^2$ oraz podpisz osie współrzędnych. Zastosuj do kolorowania algorytm pm3d.

Zadanie 4

Napisz skrypt, który wygeneruje cztery pliki postscriptowe, a w nich wizualizację aproksymacji Padego funkcji $y = e^{-x}$ w postaci słupkowej, schodkowej, impulsowej i punktowej.

Zadanie 5

Napisz program, który wylosuje 100 tysięcy liczb całkowitych z zakresu [1..100]. Teoretycznie rzecz biorąc każda liczba powinna wypaść około 1000 razy. Zapisz do pliku ile razy wypadły poszczególne liczby, wyniki przedstaw w postaci wykresu słupkowego. W ten sposób wykonasz pierwszy test generatora liczb pseudolosowych.

Zadanie 6

W fizyce potencjał elektryczny jest wielkością skalarną. Jego wartość obliczamy z wzoru V(x, y) = -kQ/r, gdzie Q to wartość ładunku elektrycznego, a r stanowi odległość punktu o współrzędnych (x, y) od tego ładunku. Dla ustalenia uwagi wartość stałej k = 1. Z punktu widzenia postawionego problemu jednostki również nie są istotne. Rozważmy świat o wymiarach 800 × 600. W świecie tym połóżmy w losowo wybranych punktach pięć ładunków elektrycznych o wartościach $Q_1 = 350, Q_2 = 400, Q_3 = 450, Q_4 = 500, Q_5 = 550$. Napisz program generujący pliki, w których zapisano wartość potencjału elektrycznego w każdym punkcie świata. Uruchom program 30 razy, a następnie stwórz postscriptową wizualizację każdego pliku w postaci dwuwymiarowej mapy pm3d (w tym celu w odpowiednim miejscu wykonaj polecenie **set pm3d map**). Wskazówka: Potencjał w każdym punkcie świata jest sumą potencjałów pochodzącą od poszczególnych ładunków.

Rozdział 10

System składu LAT $_{E}X$

| 10.1. | Wprowadzenie | 156 |
|-------|--|-----|
| 10.2. | Instalacja systemu ${\rm IAT}_{\rm E}{\rm X}$ w Ubuntu | 157 |
| 10.3. | Struktura prostego dokumentu | 158 |
| 10.4. | Wyrażenia matematyczne | 161 |
| 10.5. | Podsumowanie | 162 |
| 10.6. | Zadania | 163 |
| | | |

10.1. Wprowadzenie

System składu publikacji naukowych i technicznych LATEX jest doskonałym narzędziem służącym składaniu profesjonalnie wyglądających publikacji i to nie tylko z zakresu nauki i techniki. Dokumenty stworzone w systemie LATEX charakteryzują się ekstremalnie wysoką jakością typograficzną. Z środowiskiem LATEX nierozerwalnie związany jest system T_EX .

System T_FX został stworzony przez Donalda E. Knutha (Rys. 10.1), który prowadził nad nim prace od 1977 roku. Celem projektu było dostarczenie oprogramowania służącego generowaniu wysokiej jakości dokumentów, w sytuacji stałego pogarszania się jakości typograficznej. Knuth obserwował na przykładzie swoich własnych prac naukowych jak ważny jest profesjonalny skład w poligrafii cyfrowej. Pierwsza wersja T_FX została udostępniona w roku 1982 (niewielka poprawka pojawiła się w 1989.). T_FX to program niesłychanie stabilny, można przyjąć że wolny od błędów. Donald Knuth za wykrycie pierwszego błędu w jego systemie zaoferował nagrodę jednego centa. Za kolejny płacił już dwa centy, następnie cztery i tak dalej. W chwili obecnej nagroda przekroczyła dwieście dolarów, tym nie mniej niesłychanie trudno znaleźć jest kolejne błędy. Dla ogromnej rzeszy informatyków czek podpisany przez Knutha stanowiłby największy zaszczyt. Poszczególne wersje systemu TFX Knuth oznacza kolejnymi cyframi rozszerzenia liczby π i tak na te chwile mamy T_FX w wersji 3.141592. System edycji fontów METAFONT, który jest częścią systemu T_FX oznacza się kolejnymi rozszerzeniami liczby e (obecna wersja to 2.71828) Jako ciekawostkę można też podać, że słynący z poczucia humoru Donald Knuth zrezygnował z korzystania z poczty elektronicznej stwierdzając, że 15 lat korzystania z e-maila wystarczy na jedno życie. Po śmierci autora, zgodnie z jego wolą, zarówno TFX jak i METAFONT przestaną być rozwijane, a wszelkie znalezione błędy będą określone mianem właściwości systemu.

System LAT_EX to zestaw instrukcji zawierający polecenia, makrodefinicje i makra umożliwiające użytkownikowi tworzenie dokumentów na najwyższym poziomie z wykorzystaniem systemu T_EX. Opracowany przez Lesliego Lamporta zawdzięcza mu pierwszy człon swojej nazwy.

Czytelnik powinien zapamiętać, że słowo TEX wymawia się jak "tech", LATEX jak "latech". Wymowa wiąże się z podobieństwem litery X do greckiej litery χ - "chi". Zatem mamy tech jak technologia i to było zamierzeniem Knutha.



Rysunek 10.1: Donald E. Knuth

10.2. Instalacja systemu $\LaTeX\ensuremath{\mathrm{T}_{\mathrm{E}}}\ensuremath{\mathrm{X}}$ w Ubuntu

Istnieje wiele sposobów i opisów systemu IATEX w rozmaitych systemach operacyjnych. Chociaż TEX i IATEX występują w wersjach dla wielu systemów operacyjnych to najprzyjemniej pracuje się w nich korzystając z dobrodziejstw Linuksa. W pełni zainstalowany TEX w komputerze zajmuje kilkaset megabajtów. Z niektórych pakietów jednak większość czytelników przynajmniej na początku nie będzie korzystać.

Najprostszą metodą zainstalowania T_EXa i LAT_EXa w systemie Ubuntu 10.04 LTS będzie wykonanie dwóch poleceń:

sudo aptitude install texmaker
sudo aptitude install texlive-lang-polish

Pierwsze z nich instaluje bardzo funkcjonalny i jednocześnie wygodny edytor Texmaker służący do tworzenia nawet zaawansowanych dokumentów. Drugie zaopatruje nas w pakiet obsługi języka polskiego i polskich znaków, na wszelki wypadek, żebyśmy nie mieli z nim potencjalnych kłopotów.

Po wykonaniu tych poleceń należy uzbroić się w cierpliwość, gdyż Ubuntu zajmie się ściąganiem pakietów z repozytoriów i instalacją wszystkiego co potrzeba do poprawnego uruchomienia środowiska. W ten sprytny sposób - instalując Texmakera - zmusiliśmy system operacyjny do zainstalowania sprawnie działającego IATEXa bez konieczności instalacji od kilku do kilkudziesięciu pakietów każdy z osobna. Po zakończeniu operacji można sprawdzić czy Texmaker uruchamia się. Wykonując polecenie:

texmaker

powinniśmy otworzyć okno edytora podobne do tego jak na Rys. 10.2.



Rysunek 10.2: Edytor Texmaker - okno główne

Chociaż najlepszym edytorem do wszystkich zadań specjalnych prawdziwego hakera jest Emacs to Texmaker będzie bardzo dobry dla początkujących użytkowników ze względu na system menu widoczny po lewej stronie na Rys. 10.2. Z menu możemy uzyskać dostęp do większości najpopularniejszych poleceń, funkcje automatycznego uzupełniania składni i wiele innych. Należy jednak pamiętać, że edycję dokumentów T_EX i IAT_EX można prowadzić nawet w zwykłym notatniku.

W tym rozdziale dokumenty IAT_EXbędą tworzone w edytorze Emacs (nie będzie to miało znaczenia jeśli chodzi o ich zawartość), a przetwarzanie źródeł zapewnimy z poziomu konsoli systemu Linux.

10.3. Struktura prostego dokumentu

Każdy dokument złożony w systemie LATEX powinien charakteryzować się przejrzystą strukturą kodu źródłowego. Pamiętajmy, że LATEX to nie jest edytor tekstu podobny do Microsoft Word. To w ogóle nie jest edytor! To system składu, w którym źródło dokumentu przypomina kod napisany w specyficznym języku programowania, a dopiero odpowiednik procesu kompilacji przygotuje dla nas elegancki plik PDF. Zatem do dzieła!

Prosty dokument zapiszemy w pliku **dokument.tex** w sposób następujący:

```
\documentclass{article}
```

%Zapewnienie obsługi grafiki
\usepackage{graphicx, epsfig, epstopdf}

```
% Zapewnienie obsługi języka polskiego
\usepackage[utf8]{inputenc}
\usepackage{polski}
\usepackage[polish]{babel}
```

\begin{document}

Piękno róży przyćmiło moją jaźń.

PIĘKNO RÓŻY PRZYĆMIŁO MOJĄ JAŹŃ!

 $\end{document}$

W pierwszej linii informujemy IAT_EXa, że konstruowany dokument będzie miał formę artykułu (może być jeszcze na przykład książka **book**, raport **report** i inne). W kolejnej linii tak na wszelki wypadek informujemy system, iż możemy zażyczyć sobie wstawiania grafiki w tworzony dokument. Następnie zapewniamy poprawną obsługę języka polskiego. Właściwa treść dokumentu znajduje się między instrukcjami **begin{document}** i **end{document}**. Komentarze w pliku źródłowym rozpoczynamy od znaku %.

W celu stworzenia pliku PDF z kodu źródłowego zawartego w pliku **dokument.tex** należy wydać polecenie:

pdflatex dokument.tex

i obejrzeć produkt końcowy wykonując:

```
evince dokument.pdf
```

gdzie \mathbf{Evince} jest domyślną systemową przeglądarką plików z tym rozszerzeniem.

Póki co stworzony dokument jest bardzo prosty. Wielopoziomową strukturę nadajemy korzystając z instrukcji \section{}, \subsection{} oraz \subsubsection{}.

Wypełnienie ciała dokumentu następującą treścią:

```
\begin{document}
{
   \section{Imię Róży}
{
    Piękno róży przyćmiło moją jaźń.
   PIĘKNO RÓŻY PRZYĆMIŁO MOJĄ JAŹŃ!
   \subsection{Powitanie}
   {
    Hello World!
   \subsubsection*{Zamiast ''Lorem ipsum''}
   {
    Nie ma systemu prócz GNU, a~Linux jest jednym z~jego jąder.
}}}
```

doprowadzi do produktu końcowego jak na Rys. 10.3. Zauważmy, że ${\rm IAT}_{\rm E} X$ sam dba o numerowanie rozdziałów i podrozdziałów wszystkich poziomów.



Rysunek 10.3: Okno Evince z produktem końcowym

Prosty sposób na wstawienie dowolnej grafiki w dokumencie ilustruje fragment kodu:

```
\begin{figure}[htbp]
\centering
\includegraphics[width=10.35cm, angle=270]{wykres.eps}
\caption{Wykres funkcji dwóch zmiennych.}
```

\label{wykres1} \end{figure}

gdzie na uwagę zasługuje możliwość podania wymiaru rysunku (zarówno wysokość jak i szerokość, w centymetrach oraz w calach) oraz kąta, o jaki w razie potrzeby powinien być obrócony, a także podpisu **caption** i etykiety **label**, do której można odwoływać się bezpośrednio z tekstu przy pomocy dyrektywy $\ref{wykres1}$. IATEX sam zadba o poprawną numerację rysunków, podobnie zresztą jak o bibliografię, do której możemy się odwoływać przy pomocy dyrektywy $\cite{}$.

10.4. Wyrażenia matematyczne

Prawdziwą potęgę i moc IATEX pokazuje zawsze gdy chcemy tworzyć dokumenty zawierające skomplikowane wzory matematyczne. Nie ma na świecie edytora równań, który w tak szybki i elegancki sposób umożliwiał edycję nawet najbardziej skomplikowanych formuł. Wszystkie symbole matematyczne, fizyczne, greckie litery, operatory, nawiasy oraz inne struktury dają się zapisać przy pomocy odpowiednich poleceń IATEXa. Ograniczona objętość skryptu nie pozwala na całościowe ich przytoczenie, większość można znaleźć w darmowych i dostępnych w Internecie podręcznikach [30] i [31].

Każdą formułę matematyczną, którą chcemy umieścić bezpośrednio w tekście wstawiamy między dwa znaki \$. Jeżeli zatem w kodzie źródłowym dokumentu IAT_EX pojawi się na przykład:

\$ P = \pi r^{2} \$

będzie to oznaczało, że w dokumencie wyjściowym otrzymamy $P=\pi r^2$. Tu zauważmy, że indeks górny w wyrażeniu matematycznym (jakże pomocny przy zapisywaniu wykładnika potęgi) uzyskujemy używając znaku daszka. Analogicznie do zapisu indeksu dolnego użyjemy znaku podkreślenia. Zauważmy też, że poszczególne fragmenty formuły grupujemy w nawiasy klamrowe.

Jeżeli chcemy wstawić w tekst duży wzór - wypada uczynić to w nowym akapicie. Wtedy używamy polecenia **begin{displaymath}** i **end{displaymath}**.

I tak zapis:

```
\begin{displaymath}
a^{2}+b^{2}=c^{2} \rightarrow \
{a^{2} \over b^{2}} + 1 = {c^{2} \over b^{2}}.
\end{displaymath}
```

daje w dokumencie:

$$a^{2} + b^{2} = c^{2} \rightarrow \frac{a^{2}}{b^{2}} + 1 = \frac{c^{2}}{b^{2}}.$$

Jeżeli chcemy, żeby I^AT_EX automatycznie numerował formuły - używamy **begin{equation}** i **end{equation}**.

Podobnie zatem:

\begin{equation}
E = m c^{2}.
\label{einstein}
\end{equation}

spowoduje wyświetlenie wzoru Einsteina z numerem:

$$E = mc^2. (10.1)$$

W obu przytoczonych przykładach należy zwrócić uwagę na znaczenie pozycji nawiasów klamrowych. Pojedynczy znak \ służy do przełamywania linii jeśli zapis formuły jest zbyt długi. Poza tym, aby uniknąć błędów w kompilacji nie powinno się wstawiać pustych linii bezpośrednio nad i pod zapisem treści formuły. Do równań numerowanych typu 10.1 możemy dodawać etykiety i odwoływać się w taki sam sposób jak do rysunków.

10.5. Podsumowanie

W tym rozdziale wyjaśniono absolutne podstawy podstaw systemów składu T_EX i I^AT_EX. Podstawy te wystarczą jednak większości czytelników do podjęcia prób tworzenia własnych, bardziej zaawansowanych dokumentów. Dobrą metodą wypróbowaną przez autora jest douczanie się I^AT_EXa na bieżąco, w miarę wzrastających potrzeb. Przydatne z pewnością okaże się rozwiązanie załączonych do rozdziału zadań. Podczas poszerzania wiedzy warto korzystać z podręczników [30] i [31]. Nieocenioną pomocą są też Google i ogólnie rozumiany Internet.

W tym miejscu mogą też pojawić się pytania: Dlaczego warto używać systemu LATEX?, Dlaczego męczyć się z kompilacją kodów źródłowych dokumentów w sytuacji gdy mamy Open Office i Microsoft Word? Odpowiedzi wydają się być oczywiste. Dokumenty stworzone w LATEXu wyglądają "inaczej" - mają w sobie "to coś", ten specyficzny rodzaj elegancji i już na pierwszy rzut oka widać, że tworzone są przez profesjonalistów. Znikają również wszelkie problemy wydawnicze - wciąż zdecydowana większość czasopism naukowych z tak zwanej "górnej półki" akceptuje dokumenty tylko i wyłącznie zredagowane w LATEX. Konieczność kompilacji i ascetyczny kod źródłowy dokumentu tylko pozornie stanowią trudność. Już po kilku dokumentach zredagowanych w ten sposób można przekonać się, że w rzeczywistości praca z nimi jest szybsza i przyjemniejsza.

Podsumowując - znajomość filozofii systemu LaTeX zawsze stanowi dodatkowy atut w ręku profesjonalnego informatyka 21. wieku.

10.6. Zadania

Zadanie1

Stwórz w I^AT_EX wielopoziomowy dokument zawierający trzy sekcje, z których każda posiada dwie podsekcje i przynajmniej jedną pod-podsekcję. Wypełnij go treścią *Lorem ipsum*:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Proin nibh augue, suscipit a, scelerisque sed, lacinia in, mi. Cras vel lorem. Etiam pellentesque aliquet tellus. Phasellus pharetra nulla ac diam. Quisque semper justo at risus. Donec venenatis, turpis vel hendrerit interdum, dui ligula ultricies purus, sed posuere libero dui id orci. Nam congue, pede vitae dapibus aliquet, elit magna vulputate arcu, vel tempus metus leo non est. Etiam sit amet lectus quis est congue mollis. Phasellus congue lacus eget neque. Phasellus ornare, ante vitae consectetuer consequat, purus sapien ultricies dolor, et mollis pede metus eget nisi. Praesent sodales velit quis augue. Cras suscipit, urna at aliquam rhoncus, urna quam viverra nisi, in interdum massa nibh nec erat.

Znajdź w Internecie jak wygenerować spis treści.

Zadanie 2

Utwórz dokument $\mathbb{IAT}_{\mathbf{E}}X$, w którym zapiszesz podstawowe równania mechaniki kwantowej - równania Schroedingera.

$$\hat{H} \mid \psi(t) \rangle = i \ \hbar \frac{\partial}{\partial t} \psi(t) \rangle,$$

$$\hat{H}\psi(r,t) = i \ \hbar \frac{\partial}{\partial t}\psi(r,t).$$

Zadanie 3

Utwórz dokument $\mathbb{I}_{E}X$, w którym zapiszesz równanie Hodgkina-Huxleya opisujące działanie elektryczne fragmentu neuronu.

$$C\frac{dV}{dt} = I_{inj} - \bar{g}_{Na}m^{3}h(V - V_{Na}) - \bar{g}_{K}n^{4}h(V - V_{NK}) - \bar{g}_{L}(V - V_{L}).$$

Zadanie 4

Równania Maxwella to podstawowe równania elektromagnetyzmu. Ludwig Boltzmann napisał o równaniach Maxwella: "Czy to był Bóg, co napisał te wiersze...". W późniejszych czasach J. R. Pierce w rozdziale książki, zatytułowanym "Cudowne równania Maxwella" napisał: "Każdemu, kogo interesują nie tylko codzienne praktyczne sprawy, warto jest wyjaśnić sens równań Maxwella po prostu dla dobra jego duszy".

Utwórz dokument LATEX, w którym zakodujesz Równania Maxwella:

$$\begin{split} \oint_{L} \vec{E} \cdot d\vec{l} &= -\frac{d\Phi_{B}}{dt}, \\ \oint_{L} \vec{H} \cdot d\vec{l} &= I + \frac{d\Phi_{D}}{dt}, \\ \oint_{S} \vec{D} \cdot d\vec{s} &= \int_{V} \rho \cdot dV, \\ \oint_{S} \vec{B} \cdot d\vec{s} &= 0 \end{split}$$

oraz ich różniczkową postać:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t},$$
$$\nabla \times \vec{H} = \vec{j} + \frac{\partial \vec{D}}{\partial t},$$
$$\nabla \cdot \vec{D} = \rho, \ \vec{B} = \mu \vec{H},$$
$$\nabla \cdot \vec{B} = 0, \ \vec{D} = \varepsilon \vec{E}.$$

Zadanie 5

Utwórz dokument LATEX, w którym zapiszesz transformacje Lorentza potrzebne do wyprowadzenia równań Szczególnej Teorii Względności Einsteina:

$$t \to t' = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} (t - \frac{v}{c^2}x),$$
$$x \to x' = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} (x - vt).$$

Zadanie 6

Kiedy nadejdzie Twoja kolej napis
z pracę licencjacką, magisterską a może i doktorską w systemie
 ${\rm IAT}_{\rm E}{\rm X}.$
Rozdział 11

System tworzenia prezentacji Beamer

| 11.1. Wprowadzenie | 168 |
|--------------------------------------|-----|
| 11.2. Szkielet prezentacji | 168 |
| 11.3. Formatowanie tekstu | 171 |
| 11.4. Bloki, listy, tabele i rysunki | 171 |
| 11.5. Podsumowanie | 176 |
| 11.6. Zadania | 176 |

11.1. Wprowadzenie

Beamer jest klasą systemu LATFX służącą do przygotowywania profesjonalnych i zaawansowanych prezentacji. Prezentacje te \mathbf{sa} przeznaczone do wyświetlania z projektora multimedialnego lub drukowania na folii. Tworzenie prezentacji w Beamerze wyglada inaczej aniżeli w programach typu WYSIWYG typu Open Office Impress albo Microsoft PowerPoint i tak naprawde przypomina napisanie zwykłego dokumentu w środowisku LATEX. Komplikacja istoty rzeczy w postaci wprowadzania składni IATFX do prezentacji jest tylko pozorna. Okazuje się, że prezentacje wygenerowane z wykorzystaniem klasy Beamer wyglądają schludnie, schematy kolorystyczne zastosowane do ich dekoracji nadają dokumentom profesjonalny wygląd, a opanowanie podstawowych zasad projektowania slajdów sprawia, że mogą być one tworzone szybciej niż w programie PowerPoint.

W celu zainstalowania klasy Beamer w Ubuntu 10.04 LTS wykonujemy polecenie:

sudo aptitude install latex-beamer

przy założeniu, że posiadamy już zainstalowane środowisko IATEX.

11.2. Szkielet prezentacji

Prezentacja utworzona z wykorzystaniem klasy Beamer z reguły jest wpisana w następujący szablon:

```
\documentclass{beamer}
% polskie czcionki
\usepackage[polish]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
% Tytul i~informacja o~autorze
\title{Śrdowisko programisty}
\author{Grzegorz M. Wójcik}
\institute{gmwojcik@gmail.com}
% Wstawia dzisiejsza date
\date{\today}
\usetheme{Berlin}
```

```
\begin{document}
\frame{\titlepage}
\frame{\tableofcontents}
```

$\end{document}$

gdzie oprócz zapewnienia kodowania polskich znaków podajemy tytuł prezentacji, informację o autorze, a już bezpośrednio po rozpoczęciu ciała dokumentu wykonujemy polecenia odpowiedzialne za wygenerowanie strony tytułowej i planu prezentacji (w analogii do generowania spisu treści w dokumentach LATFX). Na uwagę zasługuje polecenie nakładające w tym przypadku na prezentację schemat kolorystyczny oraz odpowiednio zdefiniowany układ graficzny. Beamer umożliwia stosowanie wielu motywów i oprócz użytego tu schematu Berlin polecamy wypróbowanie następujących: Antibes, Bergen, Berkeley, Boadilla, Copenhagen, CambridgeUS, Darmstadt, Dresden, Frankfurt, Goettingen, Hannover, Ilmenau, JuanLesPins, Madrid, Malmoe, Marburg, Montpellier, PaloAlto, Pittsburgh, Rochester, Singapore.

Kompilacja prezentacji następuje po wydaniu polecenia:

pdflatex beamer-dokument.tex

gdzie beamer-dokument.tex to nazwa z jaką zapisaliśmy dokument.

Wygenerowany dokument można obejrzeć przy pomocy dowolnej przeglądarki dokumentów PDF. Na razie mamy tylko stronę tytułową i jeden pusty slajd (zarezerwowany na spis treści, której nie ma) (Rys. 11.1).

```
\begin{frame}
Tylko dwie rzeczy są nieskończone: Wszechświat \
oraz ludzka głupota, choć nie jestem pewien \
co~do~tej pierwszej.
\newline
\newline
Zwei Dinge sind unendlich, das Universum und die \
menschliche Dummheit, aber beim Universum bin \
ich mir nicht ganz sicher.
\newline
\begin{flushright}
[Albert Einstein]
\end{flushright}
\end{frame}
```

Slajdy tworzymy korzystając z polecenia $\mathbf{begin}{frame}$ oraz $\mathbf{end}{frame}$:

gdzie przy okazji przemycono cytat z Einsteina oraz sposoby przerzucania tekstu do nowych linii i wyrównywania do prawej (Rys. 11.2).



Rysunek 11.1: Strona tytułowa prezentacji Beamer z zastosowanym tematem Berlin



Rysunek 11.2: Przykładowy slajd. Motyw Copenhagen

11.3. Formatowanie tekstu

Każdą prezentację stworzoną w Beamerze można podzielić na sekcje, podsekcje w sposób analogiczny do wielopoziomowej struktury typowego dokumentu IATEX. Wtedy tytuły sekcji i podsekcji pojawią się na pierwszym slajdzie prezentacji, a jeśli motyw zastosowany do dekoracji dokumentu na to pozwala - na kolejnych slajdach w sposób automatyczny wbudują się odpowiednie nagłówki albo stopki. Podobnie i sam tekst może być formatowany w Beamerze dokładnie w taki sam sposób jak w dokumentach IATEX. Czcionkę pogrubioną uzyskamy stosując $textbf{}$, a pochyloną $textit{}$. Dostępne w Beamerze czcionki to: serif, avant, bookman, chancery, charter, euler, helvet, mathtime, mathptm, mathptmx, newcent, palatino, pifont, utopia. Czcionkę i jej rozmiar zmieniamy w preambule dokumentu poleceniami:

```
\documentclass[10pt]{beamer}
\usepackage{nazwa-czcionki}
```

gdzie ustawiono domyślny rozmiar czcionki na 10pt.

11.4. Bloki, listy, tabele i rysunki

Fragmenty tekstu na slajdach można grupować w tak zwane bloki. Cała prezentacja wtedy ładniej wygląda, w bloku możemy łączyć podobne treści albo wyróżniać te najważniejsze. Blok instrukcji znajduje się między poleceniami **begin{block}** i **end{block}**:

```
\begin{block}
\begin{displaymath}
E=mc^{2}
\end{displaymath}
\end{block}
```

Efektem wykonania przytoczonego tu kodu będzie Rys. 11.3. Tytuł slajdu uzyskano wykonując polecenie \frametitle{Wzór Einsteina} na początku kodu slajdu.

Wypunktowanie i numerację aranżujemy w sposób analogiczny do definiowania bloków przy pomocy: \begin{itemize} i \end{itemize}. Następujący kod:

```
\begin{itemize}
\frametitle{Klucz do~sukcesu}
\item Ubuntu 10.04 LTS
\item Emacs
```



Rysunek 11.3: Slajd prezentacji z blokiem tekstu. Motyw Darmstadt

```
\item \LaTeX\
\item Gnuplot
\item Beamer Class
\end{itemize}
```

spowoduje utworzenie slajdu takiego jak na Rys. 11.4. Jeżeli natomiast użyjemy zespołu instrukcji **\begin{enumerate}** i **\end{enumerate}** to zamiast punktorów uzyskamy liczby. Numerację zbudowaną w ten sposób można zagnieżdżać.

Wstawianie tabel w Beamerze na pierwszy rzut okaz może wydawać się skomplikowane, po pewnym czasie jednak okazuje się, że jest intuicyjne.

Następujący kod ilustruje wstawioną tablę o trzech wierszach i trzech kolumnach:

```
\begin{tabular}{|c|c|c|}
\hline
\textbf{L.p} & \textbf{wersja} & \textbf{nazwa} \\
hline
1 & 10.10 & Maverick Meerkat \\
hline
2 & 10.04 LTS & Lucid Lynx \\
hline
3 & 9.10 & Karmic Koala \\
```



Rysunek 11.4: Slajd prezentacji zawierający punktowanie. Motyw CambridgeUS

```
\hline
4 & 9.04 & Jaunty Jackalope \\
hline
5 & 8.10 & Interpid Ibex \\
hline
6 & 8.04 LTS & Hardy Heron \\
hline
\end{tabular}
```

gdzie należy zwrócić szczególną uwagę na symbole kończenia linii (podwójny backslash), separacji komórek (ampersand) i linii poziomych **\hline**. Sama definicja tabeli zbudowana jest z pionowych kresek oddzielających kolejne litery "c jak column". Efektem wykonania kompilacji slajdu z takim kodem jest Rys. 11.5.

Wstawianie rysunków w klasie Beamer zrealizowano w taki sam sposób jak w zwykłym ${\rm IAT}_{\rm E} {\rm Xu}.$ Efektem przetworzenia kodu:

```
\begin{figure}
\centering
\includegraphics[scale=0.12]{einstein.jpg}
\caption{Albert Einstein w~1947 r}
\end{figure}
```

| | | | | Srdowisko programisty | | |
|------|---------------------------|----------------------------------|----------------|-----------------------|--------------------------|----|
| Plik | Edycja Wi | dok Przejdź | Pomoc | | | |
| | Poprzednia | 🖶 Dalej | 6 (6 z 8) Dopa | suj 🔻 | | |
| | Śrdowi └─ Forr └─ T | sko progra natowanie abele | misty | | | |
| | Ubı | untu 2 | 2008-2010 | | | |
| | | L.p. | wersja | nazwa | | |
| | | 1 | 10.10 | Maverick Meerkat | | |
| | | 2 | 10.04 LTS | Lucid Lynx | | |
| | | 3 | 9.10 | Karmic Koala | | |
| | | 4 | 9.04 | Jaunty Jackalope | | |
| | | 5 | 8.10 | Interpid Ibex | | |
| | | 6 | 8.04 LTS | Hardy Heron | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | ▲日 ▲ 御 ▶ ▲ 恵 ▶ ▲ 恵 → ⑤ � | .e |

Rysunek 11.5: Slajd prezentacji z tabelą. Motyw Montpellier

będzie slajd z Rys. 11.6.

Bardzo przydatną funkcją w tworzeniu prezentacji w Beamerze jest możliwość zamieszczania fragmentów kodów rozmaitych programów na slajdach prezentacji. Służy do tego zestaw poleceń **\begin{verbatim}** i **\end{verbatim}**:

```
\begin{frame}[fragile]
\frametitle{Hello World!}
\begin{verbatim}
#include <iostream>
int main()
{
std::cout << "Hello World!" << std::endl;
return 0;
}
\end{verbatim}
\end{frame}
```

gdzie należy pamiętać o dodaniu opcji [fragile] podczas tworzenia slajdu (patrz pierwsza linia kodu). W podobny sposób włączamy fragmenty kodu w każdy dokument IAT_EX.



Rysunek 11.6: Slajd prezentacji z zamieszczonym obrazkiem. Motyw Malmoe

| Pli | Elik Edycja <u>W</u> idok Przejdź Pomo <u>c</u> | | | |
|-----|--|--|--|--|
| | | Hello World! | | |
| | Śrdowisko programisty Grzegorz M. Wójcik | R | | |
| | Wprowadzenie Einstein Formatowanie Bloki Listy Tabele Rysunki | <pre>#include <iostream> int main() { std::cout << "Hello World!" << std::endl; return 0; }</iostream></pre> | | |
| | | < ロ> (費) (差) (差) (差) (名) の(C) | | |

Rysunek 11.7: Slajd prezentacji z zamiszczonym fragmentem kodu w C++. Motyw PaloAlto

11.5. Podsumowanie

Warto tworzyć prezentacje wykorzystując klasę Beamer w systemie LATEX. Dokumenty przygotowane w ten sposób są czytelne, przejrzyste, łatwo trafiają do odbiorcy. Przystępując do projektowania prezentacji warto przede wszystkim zastanowić sie jaki jest cel i jakie ma być przesłanie. Prezentacja służy do przekazania maksymalnie dużej wiedzy w maksymalnie czytelny sposób w ograniczonym czasie. Nie znajdziemy tu animowanych przycisków, nie doświadczymy efektu obracających się slajdów. Za to możemy mieć pewność, że treści, które zamierzamy przekazać trafia do odbiorcy niezaburzone. Beamer to także rozszerzenie możliwości systemu TFX. Nie jesteśmy już tylko ograniczeni do tworzenia eleganckich dokumentów tekstowych. Dysponujemy możliwościami przekazywania wiedzy w postaci prezentacji. A wszystko za darmo z użyciem wolnego oprogramowania. Z prezentacjami przygotowanymi w Beamerze wiąże się jeszcze jedna zaleta: zazwyczaj wyglądają inaczej niż wszystkie i sam fakt tej pozytywnej odmienności powoduje zainteresowanie przedmiotem prezentacji.

11.6. Zadania

Zadanie 1

Przygotuj szablon prezentacji Beamer, w którym w łatwy sposób (bez pamiętania wszystkich nazw) będziesz mógł zmieniać użyty motyw graficzny. Wskazówka: w systemie ${\rm IAT}_{\rm E}$ Xznak % służy do komentowania linii kodu.

Zadanie 2

Stwórz wielopoziomową prezentację zawierającą trzy sekcje i po dwie podsekcje w każdej sekcji. Możesz wykorzystać tekst "*Lorem ipsum*"do wypełnienia prezentacji treścią.

Zadanie 3

W prezentacji z Zadania 2. umieść slajd, a w nim wybrany fragment tekstu ograniczony blokiem.

Zadanie 4

W prezentacji z Zadania 2. umieść slajd, w którym zastosujesz punktowanie albo numerowanie. Ciekawostka: Jeżeli poszczególne linie

zawierające punktory lub numerację rozdzielimy poleceniem \mathbf{pause} to do odkrywania kolejnych punktów trzeba będzie wciskać spację.

Zadanie 5

Wybierz sobie jakieś interesujące zagadnienie z informatyki, matematyki lub fizyki i stwórz prezentację 6-10 slajdów na ten temat.

Zadanie 6

Kiedy przyjdzie potrzeba opracuj seminarium, wystąpienie na konferencji albo wykład wykorzystując klasę Beamer.

Zakończenie

Zamierzeniem autorów było wyczerpujące przedstawienie środowiska programisty z punktu widzenia użytkownika systemu operacyjnego Linux.

Autorzy wyrażają nadzieję, że skrypt będzie stanowił ciekawą propozycję dla studentów zarówno pierwszych lat studiów informatycznych, jak i kierunków pokrewnych, zwłaszcza, że treści w nim zawarte wyczerpują standardy wymagań określone dla wykładów, wprowadzających początkujących użytkowników w zagadnienia programowania i administracji systemami uniksowymi.

Niektóre rozdziały zostały opatrzone zestawami przykładowych zadań. Zadania te mogą być realizowane w ramach ćwiczeń laboratoryjnych. Pozwala to na komplementarne prowadzenie wykładu i laboratoriów, w oparciu o spójną bazę literaturową, bez konieczności odwoływania się do wielu różnych źródeł. Bez skryptu byłoby nie do uniknięcia biorąc pod uwagę zakres poruszanej w nim tematyki.

Pozostaje nam tylko życzyć wielu przyjemności korzystania z systemu operacyjnego Ubuntu oraz narzędzi tu opisanych. Niniejszy skrypt niech więc stanowi podręcznik codziennego użytku, służący do pogłębiania uniksowej wiedzy i osobistego rozwoju, nie tylko w pierwszym, ale i w kolejnych semestrach studiów.

Spis rysunków

| 1.1. | Ken Thompson (z lewej) i Dennis Ritchie (z prawej) | 4 |
|-------|---|-----|
| 1.2. | Richard Matthew Stallman (rms) | 9 |
| 1.3. | Linus Benedict Torvalds | 10 |
| 1.4. | Pingwin Tux | 10 |
| 1.5. | Mark Shuttleworth | 11 |
| 2.1. | Ekran wyboru języka instalacji Ubuntu | 18 |
| 2.2. | Menu wyboru na początku instalacji systemu | 18 |
| 2.3. | Potwierdzenie wyboru języka | 19 |
| 2.4. | Wybór strefy czasowej | 20 |
| 2.5. | Wybór układu klawiatury | 20 |
| 2.6. | Okno wyboru sposobu podziału dysku | 22 |
| 2.7. | Okno tworzenia nowych partycji | 22 |
| 2.8. | Tworzenie partycji podczas instalacji Ubuntu | 23 |
| 2.9. | Okno zatwierdzenia dokonanych zmian i rozpoczęcia instalacji | 23 |
| 2.10. | Tworzenie pierwszego użytkownika | 24 |
| 2.11. | Logowanie do systemu - okno GDM | 25 |
| 2.12. | Propozycja spolszczenia systemu po instalacji | 26 |
| 4.1. | Ekran startowy programu pilot | 59 |
| 4.2. | Ustawianie kodowania dla KDE3.x | 61 |
| 4.3. | Ustawianie kodowania dla KDE4.x | 62 |
| 4.4. | Ustawianie kodowania dla GNOME | 62 |
| 4.5. | Definiowanie klawiszy w programie mc | 63 |
| 4.6. | Najczęściej spotykany wygląd programu mc $\hdots\dots\dots\dots\dots$ | 64 |
| 5.1. | Ekran startowy edytora Vim | 88 |
| 5.2. | Pierwszy ekran pomocy edytora vi | 88 |
| 5.3. | Ekran pomocy edytora nano | 95 |
| 5.4. | Ekran pomocy edytora mcedit w trybie monochromatycznym | 97 |
| 5.5. | Menu kontekstowe | 99 |
| 5.6. | Konfiguracja mcedit | .00 |
| 5.7. | Tryby zapisu mcedit | .00 |

| 5.8. 5.9. | Wcinanie tekstu przez zastępowanie LF |
|--|---|
| 7.1. 7.2. 7.3. 7.4. | Logowanie programem PuTTY127Ustawienie kodowania w programie PuTTY128Logowanie programem WinSCP128Interfejs programu WinSCP129 |
| 8.1. 8.2. 8.3. | Richard Stallman jako Święty IGNUcy |
| 9.1. 9.2. | Konsola Gnuplota po pierwszym uruchomieniu $\dots \dots \dots \dots 143$ Wykres funkcji $f(x) = xsin(x)$ wykonany przy pomocy polecenia plot |
| 9.3. | Wykresy funkcji $xsin(x)$ oraz $2xcos(x) + 1$ wykonane przy pomocy polecenia plot w tym samym układzie współrzędnych 144 |
| 9.4. 9.5 | Wykresy funkcji $xsin(x)$ oraz $2xcos(x) + 1$ ograniczone prostymi $x = -2, x = 7, y = -6, y = 12 \dots 145$ Trójwymiarowy wykres funkcji $z = x^2y$ |
| 9.6. | Trójwymiarowy wykres funkcji $z = x \ y \dots \dots \dots \dots \dots \dots$ 140 Trójwymiarowy wykres funkcji dwóch zmiennych z tytułami osi współrzędnych w określonych przedziałach $x,y,z \dots \dots \dots 146$ |
| 9.7. 9.8. | Wykres funkcji $z = e^{-x^2y^2}$ z nałożonym motywem kolorów 147 Wizualizacja pliku wyjściowego z aproksymacji Padego funkcji $y = e^{-x}$. Wykreślono 2.,3. i 4. kolumnę pliku wzglodom kolumny 1 140 |
| 9.9. | Wizualizacja pliku tabliczka.dat. Wykreślono zależność kolumny 3. od 1. i 2 |
| 10.1. 10.2. 10.3. | Donald E. Knuth |
| 11.1. | Strona tytułowa prezentacji Beamer z zastosowanym tematem Berlin |
| 11.2. 11.3. | Przykładowy slajd. Motyw Copenhagen |
| 11.4. | Slajd prezentacji zawierający punktowanie. Motyw CambridgeUS |
| 11.5. 11.6. 11.7 | Slajd prezentacji z tabelą. Motyw Montpellier |
| 11.(. | Motyw PaloAlto |

BIBLIOGRAFIA

- "Top 500 supercomputer sites." http://www.top500.org/, sierpnień 2010.
- [2] "Hollywood loves linux." http://news.softpedia.com/, lipiec 2010.
- [3] "Google wprowadził zakaz windowsa." http://www.tvn24.pl/, czerwiec 2010.
- [4] W. von Hagen, Ubuntu Linux Biblia Wiedza obiecana. Gliwice: Helion, 2008.
- [5] "Slackware." http://www.slackware.org/, sierpnień 2010.
- [6] "Debian." http://www.debian.org/, sierpnień 2010.
- [7] "Fedora." http://fedora.pl/, sierpnień 2010.
- [8] "Centos." http://www.centos.org/, sierpnień 2010.
- [9] "Mandriva." http://www2.mandriva.com/pl/, sierpnień 2010.
- [10] "Opensuse." http://pl.opensuse.org/, sierpnień 2010.
- [11] "Gentoo." http://www.gentoo.org/, sierpnień 2010.
- [12] "Arch linux." http://www.archlinux.org/, sierpnień 2010.
- [13] "Pc linux os." http://www.pclinuxos.com/, sierpnień 2010.
- [14] "Ubuntu linux for human beings." www.ubuntu.com, sierpnień 2010.
- [15] "Richard matthew stallman." http://www.stallman.org/, sierpnień 2010.
- [16] "Linus torvalds." http://www.cs.helsinki.fi/u/torvalds/, sierpnień 2010.
- [17] "Mark shuttleworth." http://www.markshuttleworth.com/, sierpnień 2010.
- [18] "Gnu operating systems." http://www.gnu.org/, sierpnień 2010.
- [19] "Gnu general public license." http://www.gnu.org/licenses/gpl.html, sierpnień 2010.
- [20] S. Williams, "W obronie wolności." http://stallman.helion.pl/, sierpnień 2010.
- [21] "Die.net." http://www.die.net/, sierpnień 2010.
- [22] D. J. Barret, *Linux. Leksykon kieszonkowy*. Gliwice: Helion, 2004.
- [23] B. Barnett, "The grymoire awk a tutorial and introduction." http://www.grymoire.com/Unix/Awk.html, sierpnień 2010.
- [24] B. Barnett, "The grymoire regular expressions."

http://www.grymoire.com/Unix/Regular.html, sierpnień 2010.

- [25] B. Barnett, "The grymoire sed an introduction and tutorial." http://www.grymoire.com/Unix/Sed.html, sierpnień 2010.
- [26] R. Petersen, Programowanie w Systemie Linux. Kraków: Edition, 2000.
- [27] "Programowanie w shellu bash." http://www.dief.republika.pl/main.html, sierpnień 2010.
- [28] J. Zawodny, "Emacs beginner's howto." http://jeremy.zawodny.com/emacs/, sierpnień 2010.
- [29] T. Kawano, "Gnuplot not so frequently asked questions." http://t16web.lanl.gov/Kawano/gnuplot/index-e.html, sierpnień 2010.
- [30] M. Doob, *Lagodne wprowadzenie do T_EX-a*. Winnipeg, Manitoba: Department of Mathematics, The University of Manitoba, 1993.
- [31] T. Oetiker, "Nie za krótkie wprowadzenie do systemu latex." http://www.ctan.org/tex-archive/info/lshort/polish/, sierpnień 2010.