Zarys Linuxa

Praca wykonana przez:
Andrzeja Siódmoka, SMP II

Spis treści

„Najważniejsze założenie projektowe polega na tym, że Linux powinien przynosić frajdę."

Linus Torvaldsa

1 Wstęp

1.1 Cel pracy

Praca ta w zamierzeniu ma stanowić pomost dla studentów przechodzących z systemów operacyjnych MS-DOS czy też Windows w świat **Linuxa**. Pomost jest odpowiednim słowem ponieważ stanowi ona tylko zbiór podstawowych informacji, które zdobyłem przechodząc pomiędzy tymi systemami i które uważam za bardzo pomocne, a nieraz niezbędne w tej podróży. Droga ta rozpoczyna się otwarciem wrót Linuxa, czyli zalogowniem się w systemie i prowadzi prostą ścieżką do skarbu znajdującego się na jej końcu. Tym skarbem jest pierwszy program napisany w C i uruchomiony pod **Linuxem**. Gdy już przejdziesz tę krótką drogę będziesz mógł naprawdę zacząc poznawać niesamowity świat Linuxa w czym pomocna może być bibliografia (dołączona do pracy) oraz internet – kolebka Linuxa. Pomimo, że pracę tę można porównać do pomostu to jednak nie miała na celu przeprowadzić Czytelnika przez meandry instalacji systemu, która zresztą wraz z każda nową *dystrybucją*[1] Linuxa jest coraz bardziej podobna do tej Windowsowej. Po za tym instalacja *„pingwina"* została opisana już setki razy w wielu książkach (te które moim zdaniem najlepiej spełniają to zadanie umieściłem w bibliografii na końcu pracy), więc w tym miejscu nie mogę nic nowego dodać, mogę jedynie podkreślić aby przeprowadzać ją bardzo uważnie, szczególnie podczas ustawiania parametrów monitora (znam przypadek spalenia kineskopu).

Praca została opracowana na podstawie dystrybucji RedHat mogą więc wystąpić pewne małe rozbieżności w informacjach zawartych w niej dla innych dystrybucji Linuxa (na przykład w strukturze katalogów systemowych). Starałem się jednak „wyłowić" te miejsca i podkreślić, że opis ich dotyczy tylko dystrybucji RedHat.

Na koniec tego paragrafu zamieszczam konwencję typograficzną użytą w pracy:

Krój czcionki	**Zastosowanie**
`maszyna do pisania`	Jest stosowana w przykładach do pokazania wyników poleceń, do wskazania zmiennych środowiskowych i słów kluczowych pojawiających się w kodzie oraz poleceń edytora Emacs.
pogrubiona maszyna do pisania	Używana jest w przykładach do wyróżnienia poleceń lub innego tekstu, który powinien zostać wpisany przez użytkownika.
takie pismo	Stosowane jest do zapisu nazwisk i nazw firm.

1.2 Co to jest Linux

Linux jest to 32-bitowy system operacyjny stworzony na początku lat dziewięćdziesiątych XX wieku pierwotnie dla komputerów z procesorem 386, będący darmową (co nie znaczy niedopracowaną) odmianą Unixa, która nie zawiera ani jednej linii kodu z AT&T Unixa. Obecnie istnieje także Linux na inne, niż PC platformy sprzętowe, jak Sparc, Motorola 68xxxx, PowerPC, MIPS, AXP, DEC Alpha oraz systemy

[1] Warto pamiętać o tym, że nazwa Linux dotyczy wyłącznie jądra („serca") systemu, zaś jądro z resztą oprogramowania niezbędnego do pracy systemu określa się mianem dystrybucji. Najpopularniejsze dystrybucje to: RedHat, Debian, S.u.S.E, SlackWare, Caldera OpenLinux.

wieloprocesorowe (SMP [2]), zaś w przypadku procesorów opartych o architekturę Intela istnieje możliwość jego optymalizacji dla procesorów od 386 do Pentium Pro. Linux jako system kompatybilny z Unixem oferuje wszelkie zalety tego systemu, dostarczając jednocześnie wiele innych, których komercyjne Unixy nie posiadają. Główne cechy linuxa to:

- pełna i rzeczywista wielozadaniowość z wywłaszczaniem (co najogólniej oznacza, że błędy powstałe podczas wykonywania jednej aplikacji nie wpływają na działanie innych i nie powodują destabilizacji systemu),
- wielodostępność, a zatem możliwość pracy kilku użytkowników w jednym czasie,
- duża stabilność systemu,
- system uprawnień dla każdego użytkownika,
- możliwość wielokrotnego zalogowania się na jednym komputerze (wirtualne konsole),
- wykorzystywanie w pełni możliwości sprzętowych komputera,
- kompletna i naturalna obsługa środowiska sieciowego, w tym Internetu,
- obsługa wielu systemów plików i protokołów sieciowych,
- kod źródłowy systemu i wszystkich programów,
- graficzny interface użytkownika (X-Window System),
- kompilatory wielu języków: C, C++, Pascal, Basic, Fortran, ...
- emulacja systemów DOS, Windows i Apple Macintosh,
- bogata dokumentacja (`man pages i HOWTO`),
- dystrubucją Linuxa rządzą postanowienia Licencji Publicznej GNU[3] (ang. *General Public License*; w skrócie GPL), opracowanej przez Free Software Fundation. Licencja ta chroni prawa autorskie do programów, ale umożliwia darmową dystrybucję tych programów razem z kodem źródłowym.

Dzięki tym i innym cechom Linux znajduje coraz szersze uznanie wśród użytkowników komputerów, których liczbę szacuje się obecnie kilkanaście milionów. Również profesjonaliści doceniają Linuxa i uznają go za system dopracowany i w pełni profesjonalny. Świadczą o tym takie fakty, jak uznanie przez magazyn PC Week Linuxa 2.0 za jeden z 10-ciu najlepszych produktów 1996 roku, otrzymanie w 1997 roku przez dystrybucję RedHat nagrody na konferencji producentów oprogramowania czy przenoszenie na Linuxa oprogramowania komercyjnego (np. Mathematica, CA/Clipper czy Word Perfect).

Symbolem **Linuxa** jest pingwin na poniższej fotografii trzymany na ramieniu przez twórcę systemu Linusa Torvaldsa

[2] Tymniemniej nie chodzi tu o studia matematyczno-przyrodnicze
[3] Polskie tłumaczenie licencji GNU znajduje się na stronie http:/www.linux.org.plgpl.php

1.3 Rys historyczny

W tym rozdziale pragnę krótko przedstawić fascynującą historię rozwoju systemu, który powstał na komputerze studenta w Helsinkach, a którego obecnie używa grubo ponad 10 milionów ludzi na całym świecie. Niewykluczone, że pewnego dnia i ty dopiszesz do tej historii swoje nazwisko pracując nad rozwojem Linuxa ...

- **1965:** Bell Laboratories w połączeniu z Massachusetts Institute of Technology (MIT) i General Electric (GE) rozpoczyna prace nad opracowaniem nowego wielodostępowego systemu operacyjnego pracującego na komputerach jedno- i wieloprocesorowych, z hierarchiczną strukturą katalogów i wieloma innymi wybiegającymi w przyszłość możliwościami. System ten nazwano MULTICS.
- **1969:** AT&T rezygnuje z uczestnictwa w pracach na systemem MULTICS. Części autorów tego systemu udało się zachować minikomputer PDP-7 i kontynuować prace badawcze. Programiści Bell Laboratories, K. Thompson, D. Ritchie, R. Canaday, D. McIlroy, zaimplementowali na maszynie PDP-7 pierwszą wersję systemu operacyjnego, któremu nazwę UNIX (łac. *unicus* znaczy „jedyny“) nadał w 1973r. B. Kernighan.
- **1973:** Kod źródłowy UNIX'a jest przepisany w C, nowym języku programowania wysokiego poziomu opracowanym przez D. Ritchie'go.
- **1974:** Thompson i Ritchie publikują artykuł w *„Communications of the ACM“*. Artykuł zawiera opis nowego systemu operacyjnego o nazwie UNIX. Środowiska akademickie zauważają wielki potencjał systemu jako nowego narzędzia w nauce programowania. AT&T udziela licencji uniwersytetom na wykorzystanie systemu UNIX dla celów edukacyjnych i niekomercyjnych.
- **1975:** UNIX wraz z kodem źródłowym zostaje przekazany amerykańskim uczelniom. Od tego czasu następuje faktyczny rozwój UNIX'a – duże modyfikacje do systemu wprowadził Uniwersytet Kalifornijski w Berkeley.
- **1979:** Firma AT&T zaczęła pobierać opłaty za udostępnianie kodu źródłowego. Spowodowało to powstanie wielu nowych odmian tego właśnie systemu, takich jak BSD (Berkeley Software Distribution), SunOs, Xenix, Ultrix. Firma AT&T wprowadziła do sprzedaży System V – najbardziej popularny.
- **Lata 1980's:** Unix podstawowym systemem workstation. Powstaje X-Window System (system z interfejsem okienkowym) opracowany przez Massachusetts Institute of Technology.
- **1980:** Nawet Microsoft opracował swoją wersję Unix'a (XENIX). XENIX rozprowadzany był z pakietami BASIC'a, COBOL'a, Pascal'a.
- **1991:** Doktorant Uniwersytetu Helsinskiego Linus Torval, inspirowany systemem operacyjnym MINIX (darmową bardzo uproszczoną, napisaną przez Andrew Tanenbauma wersją Unixa dla komputerów PC) zaczął tworzyć na pracę doktorską system operacyjny na komputer oparty o procesor Intel 80386.
  - **Sierpień:** pojawia się wersja 0.01. Kod źródłowy 0.01 nie był nawet wykonalny zawierał jedynie zaczątki kodu jądra.
  - **5 października:** Linus ogłosił pierwszą „oficjalną“ wersję Linuxa o numerze 0.02 od tej pory numer Linuxa stale wzrastał asymptotycznie do liczby 1.0, czyli wersji teoretycznie kompletnej i bez błędów. Po udostępnieniu systemu w październiku, dziesięć osób ściąga go, pięć poprawia.
  - **Grudzień** Sto osób dyskutuje o nim w internecie. 10 tys. linii kodu.
- **1993:** 20 tys. osób, 100 tys. linii. Ponad setka osób pracuje nad systemem. CD-ROM-y przyspieszają dystrybucje[4] .
- **Marzec 1994:** Pojawia się wersja 1.0. 100 tys. osób, 170 tys. linii kodu. Linux może być serwerem. Powstają firmy Caldera i RedHat zajmujące się dystrybucją systemu.
- **1996:** 1.5 miliona osób, 400 tys. linii kodu. Linux może używać kilka procesorów.
- **Lipiec 1999:** Najnowsze stabilne jądro ma numer 2.210. Ponad 10 milionów uzytkowników.

[4]Dystrubucje Linuxa zajmowały zazwyczaj około 50 dyskietek. Nietrudno więc wyobrazić sobie awarię jednej z nich, co znacznie komplikowało rozpowszechnianie Linuxa

1.4 Struktura Linuxa

System **Linux** jest systemem *„warstwowym“*. Najbardziej wewnętrzna jest warstwa sprzętowa (*ang. hardware*). System operacyjny (w zasadzie jego najistotniejsza część) - jądro (ang. *kernel*) bezpośrednio komunikuje się zwarstwą sprzętową i stanowi stopień pośredni dla programów użytkownika chcących skorzystać z zasobów sprzętowych komputera. Programy użytkowe nie kontaktują się bezpośrednio z sprzętem, a jedynie z jądrem systemu. Zaletą takiego rozwiązania jest to, że dobrze napisany program jest w zasadzie w 100% niezależny od warstwy sprzętowej, a przez to można go przenieść na dowolną platformę sprzętową.

Powłoka (ang. *shell*) jest programem wczytywanym bezpośrednio, po otwarciu nowej sesji i pełni rolę interfejsu między użytkownikiem i jądrem systememu. Wpisywane przez użytkownika polecenia są poddawane interpretacji przez shell, a następnie przesyłane do jądra, które zajmuje się ich wykonywaniem. Oto niektóre powłoki dostępne w Linuxie:

sh: (*Powłoka Bourne'a*) wzięła swój nazwę od jej twórcy, Stephena R. Bourne'a pracującego w Laboratoriach Bella. Została opracowana w późnych latach siedemdziesiątych i stała się podstawowym procesorem poleceń. Na początku był to jedyny wybór. Powłoka ta nie oferuje zbyt wielu działań jakie powinna spełniać, na przykład nie umożliwia edycji wiersza poleceń.

rsh: (*Ograniczona powłoka Bourne'a*) specjalna odmiana powłoki *Bourne'a* stosowana do tworzenia środowiska użytkowego o ograniczonych możliwościach i prawach dostępu. Jest ona szczególnie przydatna dla użytkowników o niewielkich umiejętnościach i znajomości Unix'a, np. wykonujących prace biurowe.

ksh: (*Powłoka Korna*) jest bardzo popularną powłoką Unix'a. Wzięła swoją nazwę podobnie jak sh od jej autora – Davida Korna, pracującego w Laboratoriach Bella. Powstała w 1983 roku, a jej publikacja odbyła się w 1986 roku. Stanowi ona rozszerzenie *powłoki Bourne'a*.

rksh: (*Ograniczona powłoka Korna*) specjalna odmiana powłoki Korna. Przeznaczona głównie do tych samych celów co **rsh**.

bash: (*Ponowiona powłoka Bourne'a*) chyba najpopularniejsza powłoka. Jest domyślną powłoką niektórych dystrybucji Unix'a a także Linuxa. Została stworzona oraz rozpowszechniona przez organizację GNU. Jest kompatybilna z orginalną *powłoką Bourne'a*.

csh: (*Powłoka C*) jej autorem jest Bill Joy, doktorant na Uniwersytecie Kalifornijskim w Berkley. Jest szczególnie używana przez programistów Unix'a. Powłoka ta oferuje kontrolę procesów, historię poleceń oraz nadawanie nazw zastępczych.

tcsh: Powłoka stanowiąca rozszerzenie Powłoki C. Oferuje dodatkowe możliwości realizacji poleceń i nazw plików oraz niektórych funkcji edycji wiersza poleceń.

zsh: Najnowsza powłoka, kompatybilna z *powłoką Bourne'a*.

Domyślą powłoką w dystrybucji RedHat jest **bash**, który znajduje się w katalogu `\bin`. Wewnętrzne polecenia i funkcje powłoki mogą służyć użytkownikom do pisania prostych programów tzw. skryptów.

2 Podstawowe polecenia systemu Linux

Zanim przejdziemy do podstawowych komend Linuxa musisz zapamiętać, że ten system w przeciwieństwie np. do MS-DOS'a odróżnia **DUŻE i małe** litery. Teraz możemy zrobić kolejny krok.

2.1 Logowanie

Jeśli zainstalowałeś już Linuxa lub uzyskałeś konto linuxowe na jakimś komputerze, to po uruchomieniu systemu na ekranie pojawi się następujące zgłoszenie:

```
Linux login:
```

Gdy wpiszesz nazwę swojego konta, np.:

`Linux login:` **Andrzej**

pojawi się prośba o hasło:

```
Password:
```

Przypominam, że duże i małe litery są rozróżnialne więc np. Andrzej oraz andrzej to zupełnie inni użytkownicy! Logowanie pozwala odróżnić jednego użytkownika od drugiego. Dzięki temu wiele osób może jednocześnie pracować w tym samym systemie komputerowym i mieć gwarancję, że tylko oni mają dostęp do swoich plików.

2.2 Znak zachęty

Gdy już uda ci się zalogować zobaczysz znak zgłoszenia, który może wyglądać na przykład tak:

```
$
```

Znak zgłoszenia może przybierać przeróżne postacie, może zawierać nazwę którą przydzieliłeś systemowi lub nazwę katalogu w którym obecnie się znajdujesz. Cokolwiek by się nie pokazało, możesz już wpisywać polecenia.

W dalszej części pracy znak zgłoszenia symbolizować będzie `$`. Oczywiście możesz zmienić jego wygląd. Do przechowywania znaku zgłoszenia służy zmienna środowiskowa o nazwie `PS1`, zdefiniowana w pliku `/etc/profile`.Aby zmienić wygląd systemowego znaku gotowości należy zmienić wartość zmiennej `PS1` na przykład w poniższy sposób:

`$ PS1='...'` – **UWAGA!** wartość zmiennej umieszczamy pomiędzy apostrofami (')

gdzie ... oznaczają dowolny tekst lub \ i symbole specjalne, rozpoznawalne przez powłokę. Tabela zawiera listę symboli specjalnych wykorzystywanych w definicji znaku zachęty powłoki `bash`.

Symbol	Znaczenie
\!	Wyświetl numer polecenia w historii.
\#	Wyświetl numer aktualnego polecenia.
\$	Wyświetl 0, # lub $.
\W	Wyświetl podstawową nazwę aktualnego katalogu.
\\	Wyświetla ukośnik (\).
\[	Początek sekwencji niewyświetlanych znaków.
\]	Koniec sekwencji niewyświetlanych znaków.
\d	Wyświetl aktualną datę.
\h	Wyświetl nazwę hosta.
\n	Znak końca wiersza.
\nnn	Znak o numerze ósemkowym *nnn*.
\s	Wyświetl aktualną godzinę.
\u	Wyświetl nazwę użytkownika.
\w	Wyświetl nazwę aktualnego katalogu.

$ PS1='[\u@\h \W]\ $ '

W moim przypadku po tej operacji znak zgłoszenia wygląda następująco:

```
[Andrzej@localhoste Andrzej] $
```

2.3 Zmiana hasła użytkownika

Aby zmienić swoje hasło należy wpisać polecenie `passwd`. Zgłosi się ono z zapytaniem o stare hasło[5] (żeby była pewność, że to naprawdę ty), zapyta o nowe, a na końcu poprosi, abyś wpisał je drugi raz – robi to aby mieć pewność, że wpisałeś je bezbłędnie.

Istnieją zasady budowania takich haseł, które innym będzie trudno odgadnąć. Niektóre systemy sprawdzają hasła i odrzucają te, które nie spełniają minimalnych wymagań. Na przykład często hasło musi mieć co najmniej sześć znaków. Co więcej, powinny tam wystąpić pomieszane wielkie i małe litery oraz znaki inne niż litery i cyfry. Jeśli więc nie powiedzie się zmiana hasła, zwróć uwagę na powyższe wymagania i powtórz procedurę.

Po satysfakcjonującym skonstruowaniu znaku zachęty i wybraniu odpowiedniego hasła możemy zacząć podróżować po katalogach systemu.

2.4 Katalogi

Pliki podobnie jak w innych systemach umieszczone są w katalogach. Układ katalogów przypomina strukturę drzewa, czyli jest katalog główny – **korzeń**, a gałęzie to podkatalogi i pliki. Katalog główny, jest katalogiem specjalnym, tworzonym podczas instalacji Linux. Duża część katalogów to tak zwane katalogi systemowe, muszą one występować w określonej hierarchii (w zależności od dystrybucji mogą wystąpić pewne różnice w ich ułożeniu). W poniższej tabeli zawarte są bardzo skrótowe opisy niektórych katalogów Linux (dla dystrybucji RedHat). Na razie mogą wydać się niezrozumiałe, ale w przyszłości, gdy już będziesz umiał swobodnie poruszać się po katalogach, te opisy okażą się przydatne.

[5] Jeśli jeszcze nie posiadasz hasła, to oczywiście to pytanie się nie pojawi.

Katalog	Opis
/home	Zawiera katalogi domowe (osobiste) użytkowników.
/bin	Znajdują się tu standardowe polecenia, takie jak ls czy pwd.
/usr/bin	Inne polecenia. Rozróżnienie między /bin i /usr/sbin jest arbitralne.
/usr/sbin	Polecenia administracyjne używane głównie przez administratora systemu.
/lib oraz /usr/lib	Wszelkiego rodzaju biblioteki.
/usr/doc oraz /usr/share/doc	Zawiera dokumentację Linuxa oraz programów.
/etc	Zawiera pliki konfiguracyjne.
/var	Pliki administracyjne, takie jak pliki rejestracyjne, wykorzystywane przez programy użytkowe.
/var/spool	Miejsce tymczasowego przechowywania plików, np. tych które są drukowane.
/usr	Polecenia i programy dla użytkowników (zawiera wiele podkatalogów).
/dev	Pliki odpowiadające urządzeniom, takim jak dyski twarde czy CDROM.
/root	Katalog domowy administratora.
/usr/man oraz /usr/share/man	Pliki podręczników elektronicznych.
/tmp	Zawiera pliki tymczasowe.
/boot	Miejsce, w którym czasami są przechowywane jądro i inne pliki używane podczas inicjacji systemu.

Każdy użytkownik ma swój podkatalog katlogu /home/. Jeśli twoja nazwa rejestracyjna to Andrzej, to twoje pliki będą umieszczone w podkatalogu /home/Andrzej, zwanym twoim katalogim macierzystym lub domowym. Oczywiście możesz w nim utworzyć więcej podkatalogów. Jak widzisz nazwy katalogów oddzielone są ukośnikami „/“, tę podzieloną ukośnikami listę nazywa się często ścieżką. W momencie logowania system „przenosi cię“ do twojego katalogu macierzystego...

2.5 Poruszanie się wśród katalogów - polecenia pwd, oraz cd

... można to sprawdzić używając polecenia:

pwd (ang. *print working directory*)– polecenie podaje katalog roboczy (bieżący).

```
$ pwd
$ /home/Andrzej
```

Oczywiście nie musisz cały czas znajdować się w katalogu macierzystym, wyjście z niego umożliwa polecenie:

cd - (ang. *change directory*) polecenie służące do zmiany katalogu bieżącego.

```
$ cd /usr/bin
$ pwd
$ /usr/bin
```

Zwykle używa się jeszcze:

Polecenie	Znaczenie
cd ..	Przejście do katalogu nadrzędnego.
cd ~	Przejście do katalogu domowego; to samo wykonuje cd
cd [ścieżka]	Przejście do katalogu o podanej ścieżce.

2.6 Tworzenie i kasowanie katalogów – polecenia mkdir oraz rmdir

Po opanowaniu poruszania się po katalogach warto zacząć je tworzyć służy do tego:

mkdir (ang. *make directory)*– po poleceniu wystarczy podać nazwę nowego katalogu. Utwórzmy katalog, w którym w przyszłości będą znajdować się nasze programy.

```
$ mkdir /home/Andrzej/programs
```

Za pomocą `mkdir` można również szybko tworzyć całe struktury katalogów. Flaga `-p` (ang. *parent*) spowoduje utworzenie wszystkich wpisanych po niej katalogów (katalogi już istniejące nie zostaną nadpisane). Teraz możemy utworzyć katalog C (w przyszłości umieścimy w nim programy napisane w C) i podkatalog `do_wyrzucenia` (w którym chcemy przechowywać złe programy napisane w C):

`$ mkdir -p /home/Andrzej/programs/C/do_wyrzucenia`

Przeciwieństwem polecenia `mkdir` jest polecenie `rmdir`, które usuwa katalogi. Usuwane katalogi muszą być puste[6]. Ponieważ nie będziemy pisać złych programów w języku C, usuniemy pusty katalog `do_wyrzucenia`:

```
$ rmdir /home/Andrzej/programs/C/do_wyrzucenia
$ ls /home/Andrzej/programs/C
$
```

2.7 Badanie zawartości katalogu – polecenie `ls`

Polecenie **ls** (ang. *list*) służy do wyświetlania zawartości jednego lub większej ilości katalogów. Znak zachęty tuż po wywołaniu `ls` w trzeciej linijce poprzedniego przykładu oznacza, że katalog `C` jest pusty, czyli operacja kasowania katalogu `/do_wyrzucenia` powiodła się! Istnieje ponad 40 flag umożliwiających sformatowanie wyświetlanej listy. Domyślnie `ls` sortuje nazwy w porządku alfabetycznym i wyświetla w kolumnach:

```
$ ls /home/Andrzej
Desktop   Linux-praca   programs
```

Co oznacza, że w moim katalogu macierzystym znajdują się trzy katalogi: `Desktop, Linux-praca i programs` Do zmiany formatu wyświetlania służy szereg flag.

Flaga	Opis
`-d`	Wypisz katalogi pomijając pliki.
`-r`	Sortowanie plików w odwrotnej kolejności.
`-l`	Wyświetl szczegółowe informacje o każdym pliku i katalogu (ang. *long*).
`-a`	Wyświetla wszystkie katalogi i pliki, nawet te ukryte.
`-m`	Wypisz nazwy plików rozdzielone przecinkami.
`-x`	Wypisz nazwy w wierszach zamiast w kolumnach.
`-A`	Wyświetl wszystkie pliki z wyjątkiem . i .. (ang. *almost-all*).
`-C`	Wyświetl nazwy w kolumnach.
`-F`	Oznacz katalogi, dowiązania i pliki wkonywalne.
`-R`	Wypisz rekurencyjne zawartość wszystkich katalogów zagnieżdżonych (ang. *recursive*).
`-S`	Posortuj pliki według rozmiaru (ang. *sort=size*).
`-color`	Oznacz pliki odrębnym kolorem.
`-t`	Pliki zostaną wyświetlone według czasu modyfikacji, najnowsze jako pierwsze.

Jeśli jednak chcemy zobaczyć co dokładnie znajduje się w naszym katalogu domowym użyjemy flagi `-a`:

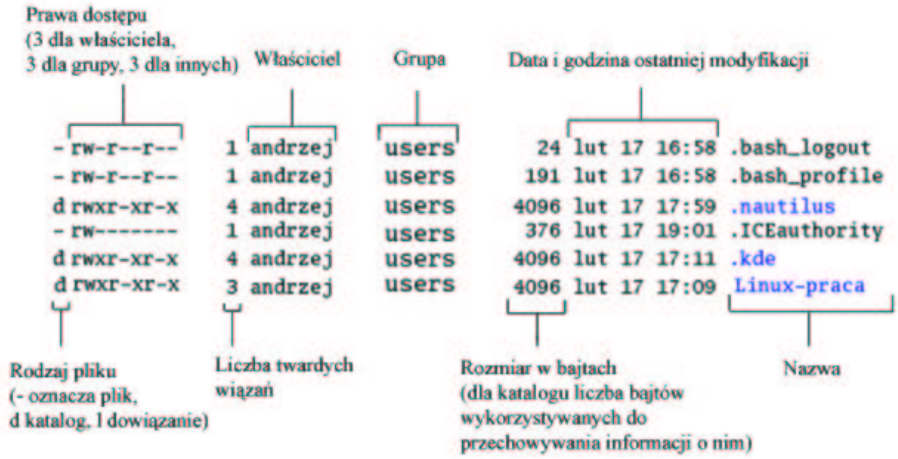
`$ ls -a /home/Andrzej`

[6] Aby usunąć katalog, który nie jest pusty należy użyć polecenia `rm -fr`. Więcej informacji o tym poleceniu znajdziesz w dalszej części pracy.

```
.                                    .emacs            .mcop
..                                   .emacs.d          .mcoprc
.autorun.lck                         .first_start_kde  programs
.bash_logout                         .gtkrc            .qt
.bash_profile                        .gtkrc-kde        .wmrc
Desktop                              .ICEauthority     .Xauthority
.DCOPserver_localhost.localdomain__0 .kde              .xftcache
.DCOPserver_localhost.localdomain_:0 Linux-praca       .xsession-errors
```

Rozszyfrujmy wynik otrzymany przez polecenie z flagą `-a`. Pojedyńcza kropka (w górnym lewym rogu) oznacza bieżący katalog, podwójna katalog nadrzędny (o jeden poziom wyżej). Wśród zwykłych katalogów i plików widzimy takie, które zaczynają się od kropki, są to tak zwane pliki ukryte (niemożliwe jest wyświetlenie ich za pomocą zwykłego `ls` – porównaj dwa ostatnie przykłady). Generalnie możesz nie pamiętać, że te pliki istnieją, ale będą bardzo przydatne, kiedy będziesz konfigurował swój system.

Niektóre flagi można ze sobą łączyć np. polecenie `ls -al` wyświetli wszystkie pliki i katalogi (nawet te ukryte) wraz ze szczegółowymi informacjami na ich temat. Poniższy rysunek przedstawia typowy rezultat tego polecenia oraz wyjaśnienie co oznacza każde pole.



2.8 Prawa dostępu

Pola z lewej strony rysunku oznaczają prawa dostępu, określają one sposób, w jaki można używać pliku. W linuxie istnieją trzy rodzaje praw dostępu do pliku:

Litera	Znaczenie
`r`	*Prawo do odczytu* oznacza, że możesz oglądać zawartość pliku.
`w`	*Prawo do zapisu* oznacza, że możesz dokonywać zmian w pliku lub go usuwać.
`x`	*Prawo do wykonywania* oznacza, że możesz uruchomić plik lub program.

Dla katalogów prawa dostępu mają inne znaczenie:

Litera	Znaczenie
`r`	*Prawo do odczytu* oznacza, że możesz wypisać zawartość katalogu.
`w`	*Prawo do zapisu* oznacza, że możesz dodawać lub usuwać pliki z katalogu.
`x`	*Prawo do wykonywania* oznacza, że możesz wypisać informację o plikach z tego katalogu.

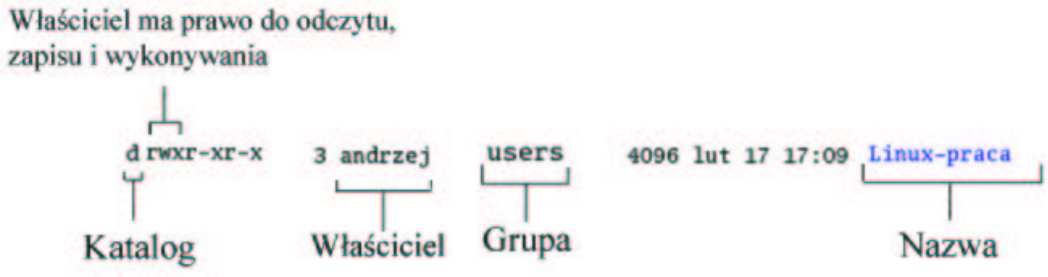
Różnica pomiędzy prawem do odczytu a prawem do wykonywania dla katalogów jest bardzo niewielka. Zazwyczaj katalogi mają przypisane oba te prawa albo żadnego z nich.

Kto może dostać prawa dostępu?

Linux, aby umożliwić ludziom wspólną pracę, posiada trzy poziomy praw dostępu. Są to: właściciel, grupa oraz inni. „Inni“ dotyczy wszystkich, którzy mają dostęp do systemu, a nie są właścicielami lub członkami grupy. Pomysł grupy wziął się stąd, żeby grono użytkowników, na przykład programistów, miało dostęp do pliku. Zapewne programista piszący kod źródłowy będzie chciał z jednej strony, zarezerwować dla siebie prawo do zapisu, z drugiej – umożliwić innym członkom jego grupy odczyt oraz sprawić by „inni“ nie mieli żadnych praw dostępu.

Tak więc każdy plik ma swojego właściciela i grupę. Właścicielem jest na ogół użytkownik, który utworzył plik. Każdy użytkownik należy do domyślnej grupy. Grupa ta jest przypisana do każdego pliku, który stworzy ten użytkownik.

Podsumowując mamy trzy prawa dostępu (do odczytu, zapisu i wykonywania) i trzy poziomy (właściciel, grupa oraz inni). Przyjrzyjmy się więc pewnemu katalogowi i popatrzmy, jakie prawa dostępu ma przypisane.



Pierwszy znak, litera `d` oznacza właśnie katalog. Następne trzy bity dotyczą właściciela. Właściciel – andrzej ma wszystkie trzy prawa dostępu. Kolejne trzy bity odnoszą się do członków grupy w tym wypadku `users`.Mają oni prawo odczytu i wykonania, ale nie posiadją prawa zapisu ponieważ w polu, które powino zawierać literę `w` znajduje się dywiz. Ostatnie trzy bity dotyczą innych, którzy mają tutaj te same prawa co grupa.

2.9 Edycja tekstu w wierszu poleceń

Po poznaniu kilku poleceń, przydatne będzie poznanie operacji na tekscie wiersza poleceń. Możliwość edycji tekstu w wierszu poleceń (czyli np. ścieżki dostępu) zależą od używanego `shella`. Domyśly `shell` dystrybucji RedHat – `bash`, dopuszcza bardzo użyteczne operacje na tych tekstach:

Operacja	**Klawisz lub kombinacja klawiszy**
Do przodu o jeden znak	Strzałka w prawo
Do tyłu o jeden znak	Strzałka w lewo
Do przodu o jedno słowo	Alt+F
Do tyłu o jedno słowo	Alt+B
Początek wiersza	Ctrl+A; Home
Koniec wiersza	Ctrl+E; End
Skasuj znak	Ctrl+D; Backspace
Skasuj słowo	Alt+D
Skasuj wszysto do końca wiersza	Ctrl+K
Skasuj wszystko do początku wiersza	Ctrl+U

Jeżeli używany `shell` obsługuje historię poleceń wówczas można, korzystając z kursorów (strzałka w górę i w dół) szybko przywoływać wpisane wcześniej polecenia.

Inną zaletą shella `bash` jest możliwość uzupełniania poleceń. Aby szybko odszukać wszystkie polecenia zaczynające się od podanego wzorca, należy wpisać pierwsze kilka liter nazwy polecenia i nacisnąć **Tab**. Przykładowo:

```
$ contr
```
po wciśnięciu **Tab**, ujrzymy następujące polecenie:
```
$ control-panel
```

Jeśli wpisane litery wskażą jednoznacznie polecenie to shell po naciśnięciu **Tab** automatycznie uzupełni resztę nazwy. Jeśli jednak istnieje kilka poleceń zaczynających się od podanych liter to trzeba nacisnąć klawisz **Tab** dwukrotnie co spowoduje wyświetlenie wszystkich możliwości uzupełnienia podanych liter:
$ pi
Po dwukrotnym naciśnięciu **Tab**, pojawi się lista:
```
 pi1toppm  pic       pick  picttoppm  pilot  ping
 pi3topbm  pic2tpic  pico  pidof      pine
```

Te kilka poleceń znacznie ułatwia i przede wszystkim skraca czas wpisywania poleceń.

2.10 Maski

Maski stosuje się w wierszu poleceń do wskazania określonych grup plików. Powłoki linuxowe udostępniają wiele różnorodnych masek, pozwalających na odnajdywanie plików według różnych wzorców. Najczęściej stosowane maski zostały umieszczone w poniższej tabeli:

Wyrażenie	**Oznacza**
`*`	Dowolny łańcuch znaków.
`?`	Dowolny pojedyńczy znak.
`[a-z]`	Znaki z określonego zakresu od a do z.
`[0123]`	Znaki z określonego zbioru - 0,1,2 lub 3.
`\?`	Znak ?
`\)`	Znak)
`^abc`	Tekst zaczynający się od abc.
`$ abc`	Tekst kończący się na abc.

Powiedzmy, że masz katalog zawierający następujące pliki źródłowe w C:

$ ls
```
prog1open.c     prog2open.c     prog3open.c     prognewopen.c     progclose.c
```

Aby zostały wypisane trzy pliki zawierające w swojej nazwie cyfry, możesz wpisać:

$ ls prog?open.c
```
prog1open.c     prog2open.c     prog3open.c
```

Powłoka szuka jednego znaku zastępującego znak zapytania. Zatem wyświetli `prog1open.c, prog2open.c, prog3open.c`, ale nie `progclose.c`, ponieważ nazwa zawiera za dużo znaków.

Jeśli nie interesuje cię drugi plik możesz wskazać tylko te, których chcesz używać za pomocą nawiasów kwadratowych:

$ ls prog[13]open.c
```
prog1open.c     prog3open.c
```

Jeśli któryś znak umieszczony w nawiasach kwadratowych pasuje do nazwy pliku, to zostanie ona wyświetlona. W nawiasach kwadratowych możesz podać także zakres znaków:

$ prog[1-3]open.c
```
prog1open.c     prog2open.c     prog3open.c
```

W ten sposób znowu wyświetliliśmy trzy pliki. Dywiz oznacza „dopasuj" każdy znak z przedziału od 1 do 3. Możesz prosić o dowolny znak numeryczny, wpisując `[0-9]` lub o dowolny znak alfabetu – za pomocą `[a-zA-Z]`. W ostatnim przykładzie potrzebne są dwa zakresy, gdyż powłoka rozróżnia wielkie i małe litery.

Załóżmy, że chcesz zobaczyć także plik `prognewopen.c`. Możesz teraz użyć gwiazdki, ponieważ chcesz dopasować każdą liczbę znaków występującą pomiędzy ciągami `prog` a `open`:

```
$ ls prog*open.c
prog1open.c     prog2open.c     prog3open.c     prognewopen.c
```

Obecnie gwiazdka oznacza „żaden znak, jeden lub kilka znaków", więc jeśli istniałby plik `progopen.c`, to także zostałby pokazany.

2.11 Tworzenie i uaktualnianie plików – polecenie `touch`

Polecenie `touch` służy do uaktualniania i tworzenia plików w Linuxie. Aby utworzyć nowy plik należy wydać polecenie:

```
$ touch /home/Andrzej/programs/C/skarb.c
```

spowoduje utworzenie pliku o zerowej wielkości i nazwie skarb.c . Linuxowe nazwy plików mogą składać się aż z 256 znaków oraz zawierać szereg znaków specjalnych. Istnieją jednak pewne znaki, których nie można stosować w nazwach plików. Są to miedzy innymi:

```
" , ' * & ) ( | ! ` ? \ / < > ;
```

Możliwe jest utworzenie kilku plików jednocześnie, przykładowo wydając polecenie:

```
$ touch /home/Andrzej/programs/C/program.c tekst.tex skasuj_mnie.txt
```

utworzone zostaną trzy pliki o nazwach: `program.c, tekst.tex oraz skasuj_mnie.txt`. W przypadku użycia polecenia na istniejącym pliku, zostanie zmieniona godzina i data jego modyfikacji na bieżącą. Polecenie touch, może być użyte z opcjami przedstawionymi w tabeli poniżej:

Flaga	Opis
`-c`	Nie istniejące pliki nie będą tworzone.
`-a`	Zmiana czasu ostatniego dostępu do pliku.
`-m`	Zmiana tylko czasu ostatniej modyfikacji pliku.

Tworzyć pliki oraz je modyfikować można także za pomocy edytorów, takich jak na przykład *vi* czy **Emacs** (temu ostatniemu poświęcimy osobny rozdział).

2.12 Usuwane plików i katalogów – polecenie `rm`.

Skoro umiemy już tworzyć pliki warto nauczyć się je kasować (jeden plik zresztą sam się o to prosi). Zadanie umożliwia nam polecenie `rm`:

```
$ cd /home/Andrzej/programs/C
$ rm skasuj_mnie.txt
$ ls
```

Flagi dla polecenia rm:

Flaga	Opis
`-r`	Użyta wraz z nazwą katalogu powoduje usunięcie wszystkich plików w tym katalogu oraz jego samego.
`-i`	Przed usunięciem pliku, `rm` prosi o potwierdzenie.
`-rf`	Połączenie tych opcji spowoduje skasowanie całej struktury wybranego katalogu.

Polecenie `rm` dopuszcza użycia masek, na przykład polecenie:

```
$ rm prog*
```

usunie wszystkie pliki zaczynające się od ciągu znaków `prog`.

Polecenie `rm` może również służyć do kasowania katalogów, ale trzeba w tym celu skorzystać z flagi `-r` oraz `-f`. Połączenie tych opcji spowoduje skasowanie całej struktury wybranego katalogu bez prośby o potwierdzenie.

Ostrzeżenie:

Polecenie rm należy używać bardzo ostrożnie, szczególnie z flagą -rf, ponieważ powoduje nieodwracalne usunięcie plików (katalogów).

W przykładzie z rozdziału 2.6, skasowaliśmy katalog `do_wyrzucenia` przy pomocy `rmdir` możemy uzyskać ten sam rezultat w poniższy sposób:

$ rm -rf /home/Andrzej/programs/C/do_wyrzucenia

2.13 Aliasy

Dobrym nawykiem jest zawsze używanie `rm` wraz z flagą `-i`. Zapewnia to wiekszą pewność, że nie usuniemy przypadkowo cennego pliku. Aby nie musieć pamiętać o fladze `-i` możemy wykorzystać aliasy. Aliasy są to skrótowe nazwy często używanych poleceń w Linuxie. Polecenie to jest wbudowane w powłokę **bash** jak rówmież **csh**. Za pomocą aliasów możesz zastąpić długie polecenia, krótkimi i łatwymi do zapamiętania skrótami. Aby utworzyć alias musisz wydać polecenie o następującej składni:

$ alias twój_skrót='polecenie'.

Tak więc aby zabezpieczyć się przed przypadkowym skasowniem pliku, stworzymy następujący alias:

$ alias rm='rm -i'

Korzystanie z tego aliasu zapewnia każdorazowe pytanie czy na pewno chcemy skasować dany plik. W celu zlikwidowania zdefiniowanego aliasu, musisz wydać polecenie:

$ unalias twój_skrót

Polecenie `alias` bez jakichkolwiek opcji wyświetla listę aktualnie zdefiniowanych skrótów.

2.14 Kopiowanie plików i katalogów – polecenie cp

Umiejętność kopiowania plików i katalogów jest nieodzowna dla każdego użytkownika Linuxa.

Kopiowanie plików

Polecenie `cp` (ang. *copy*) służy do kopiowania plików. Aby nadać kopii wskazanego pliku nową nazwę, należy po `cp` wypisać obie nazwy (istniejącą i nową):

$ cp plik1 plik2

Na przykład jeśli chcemy skopiować plik `program.c` do katalogu macierzystego i nazwać go programik.c napiszemy:

```
$ cp /home/Andrzej/programs/C/program.c /home/Andrzej/programik.c
$ ls /home/Andrzej
Desktop   Linux-praca   programs   programik.c
```

Można użyć znaku „~“ by zastąpić ścieżkę do katalogu domowego, tzn. powyższe polecenie można zapisać następująco:

```
$ cp ~/programs/C/program.c ~/programik.c
```
Ponieważ `cp` dopuszcza użycie masek więc aby skopiować pliki, których nazwa zaczyna się od liter `plik` do wybranego katalogu, należy wpisać:

$ cp plik* /tmp

Przy korzystaniu z `cp` należy zachować ostrożność. O ile nie użyje się flagi -i, wszystkie pliki o nazwach pokrywających się zostaną nadpisane bez ostrzeżenia. Jeśli jednak skorzystamy z flagi `-i`, zostaniesz każdorazowo ostrzeżony przed nadpisaniem istniejącego pliku:

```
$ cp -i plik1 plik2
cp: overwrite 'plik2'? y
```

Jeszcze bezpieczniejsze jest zastosowanie flagi `-b`. Powoduje ona automatyczne utworzenie kopii zapasowych wszystkich nadpisywanych plików.

```
$ cp -bi plik1 plik2
cp: overwrite 'plik2'? y
$ ls plik2*
plik2 plik2~
```

Nazwy kopii zapasowych kończą się tyldą (~).

Kopiowanie katalogów

Połączenie flag -P i -R spowoduje skopiowanie nie tylko plików znajdujących się w danym katalogu, ale również całej zagnieżdżonej w nim struktury podkatalogów. Każde z poniższych poleceń kopiuje cały katalog `kat1` i wszystkie zawarte w nim pliki i podkatalogi do katalogu `kat2`:

```
$ cp -Pr kat1 kat2
$ cp -r kat1 kat2
$ cp -R kat1 kat2
```

Polecenie cp ma wiele różnych opcji (ponad 40), w tabeli poniżej zawarte są niektóre z nich:

Flaga	Opis
`-a`	Zachowuje wszystkie atrybuty oryginalnych plików.
`-b`	Tworzy kopie zapasowe plików które mają być nadpisane lub usunięte.
`-d`	Nie przerywa symbolicznych dowiązań; zachowuje dowiązanie pomiędzy źródłem i kopią.
`-i`	Pyta czy nadpisać istniejące pliki.
`-l`	Zamiast kopiować, tworzy dowiązania twarde.
`-p`	Zachowuje wszystkie informacje, włączając właściciela, grupę, prawa dostępu i kod czasu.
`-P`	Kopiuje pliki do katalogu docelowego traktując go jako macierzysty, powiela strukturę katalogów, w której były pliki źródłowe.
`-r`	Kopiuje także podkatalogi.

2.15 Przenoszenie i zmiana nazw plików i katalogów – polecenie `mv`

Polecenie `mv` (ang. *move*) służy do zmiany nazw plików i katalogów oraz do ich przenoszenia. Aby zmienić nazwę istniejącego pliku, należy po `mv` wpisać obie nazwy (kolejno obecną i nową), w przeciwieństwie do `cp`, polecenie `mv` kasuje plik źródłowy:

$ mv plik1 plik2

Zupełnie analogicznie zmienia się nazwy katalogów:

$ mv kat1 kat2

Podobnie jak w przypadku `cp, mv` nie ostrzega przed nadpisaniem istniejący plików i katalogów, chyba że skorzystamy z flagi `-i`:

```
$ mv -i plik1 plik2
mv: replace 'plik2'? y
```

Bezpieczniej jednak skorzystać z omówionej już wcześniej flagi `-b` (*kopia zapasowa*). Po połączeniu jej z flagą `-i, mv` poprosi o potwierdzenie chęci nadpisania istniejącego pliku, a w przypadku uzyskania potwierdzenia utworzy jego kopię zapasową (zupełnie analogicznie jak dla polecenia `cp`):

```
$ mv -i plik1 plik2
mv: replace 'plik2'? y
$ ls plik*
plik2 plik2~
```

Polecenie `mv` potrafi przenosić pliki do innych katalogów:

```
$ mv plik1 /tmp
$ ls /tmp/plik/
plik1
```

Jeśli chcemy jednocześnie przenieść pliki do nowego katalogu i zmienić jego nazwę, musimy podać nową nazwę w ścieżce wybranego katalogu:

```
$ mv plik1 /tmp/plik2
```

To samo tyczy się przenoszenia katalogów:

```
$ mv kat1 /tmp/kat2
```

Jeśli katalog docelowy nie istnieje, `mv` zmienia nazwę katalogu źródłowego na `kat2`, w przeciwnym wypadku cała zawartość katalogu `kat1` zostanie przeniesiona do katalogu `kat2`.

2.16 Tworzenie dowiązań symbolicznych

Dowiązanie symboliczne jest rodzajem pseudopliku, który jedynie wskazuje inny plik. Gdy chcesz edytować, czytać lub wykonywać wiązanie symboliczne, system jest na tyle sprytny, że podstawia ci prawdziwy plik. Dowiązania symboliczne działają podobnie jak skróty w systemie Windows, ale posiadają o wiele więcej możliwości. Rozpatrzmy przykład z programem `prog`. Załóżmy, że chcesz utworzyć dowiązanie o nazwie `prog` wskazujące aktualny plik, który ma nazwę prog1.c . Wpisz następujące polecenie:

```
$ ln -s prog1.c prog
```

Utworzyłeś w ten sposób nowy plik o nazwie `prog`, który jest rodzajem pseudopliku. Jeśli go uruchomisz, to tak naprawdę uruchomiony zostanie `prog1.c`. Sprawdźmy co ma do powiedzenia o tym pliku polecenie `ls -l`:

```
$ ls -l prog
lrwxrwxrwx     2     Andrzej     users     8 sep 17 14:35     prog -> prog1.c
```

Litera `l` na początku linii oznacza, że ten plik jest dowiązaniem, a strzałka `->` wskazuje rzeczywisty plik, do którego prowadzi dowiązanie.

Dowiązania symboliczne są naprawdę proste, musisz tylko przywyknąć do pomysłu, że jeden plik wskazuje na drugi.

Za pomocą polecenia `ln` można również tworzyć dowiązania do często wykorzystywanych katalogów:

$ ln -s /windows /mnt/windows

Teraz zamiast każdorazowo podawać całą ścieżkę (na przykład przy kopiowaniu plików), wystarczy wpisać:

$ cp plik.txt /windows

2.17 Znajdowanie plików i katalogów - polecenia `find` oraz `locate`

Przeglądanie drzewa katalogów w poszukiwaniu interesujących plików jest dość uciążliwe, dlatego Linux został wyposażony w szereg narzędzi automatyzujących tę czynność

Polecenie `find`

Polecenie to służy do znajdowania wybranych plików i katalogów na obszarze całego systemu plików. Potrafi wyszukiwać pliki według nazwy, rozmiaru, typu lub daty modyfikacji. Wymaga podanie ścieżki do przeszukania i wzorca nazwy pliku do odnalezienia:

$ find /usr -name pico -print -xdev

W tym wypadku szukamy w katalogu `/usr` edytora `pico` i poświęconej mu strony podręcznika. Pamiętajmy, że poszukiwania całego systemu plików razem z podmontowanymi doń systemami innych komputerów (dostępnych przez sieć lokalną) może trwać długo. Oto efekt naszych poszukiwań:

/usr/bin/pico
/usr/man/man1/pico.1.gz

Aby ograniczyć obszar przeszukiwań do lokalnego systemu plików, należy skorzystać z opcji `-xdev`. Jeśli ją pominiemy, przeszukane zostaną wszystkie systemy zewnętrzne (na przykład CD-ROM i podmontowane drzewa katalogu Windows). Aby odnaleźć nowe lub rzadko używane programy, należy skorzystać z opcji `-atime`. Przykładowo, poniższe polecenie wyszukuje programy uruchamiane przynajmniej raz w ciągu ostatnich 100 dni:

$ find /usr/bin -type f -atime +100 -print

Teraz szukamy programów mających nie więcej niż jeden dzień:

$ find /usr/bin -type f -atime -1 -print

Do znajdowania plików o podanym rozmiarze służy opcja `-size`. W tym przypadku należy podać liczbę bloków (1 blok = 512 bajtów) lub kilobajtów (1 kilobajt = 1024 bajty) zajmowany przez plik. Poniższe polecenie znajduje w katalogu `find /usr/bin` wszystkie pliki większe niż 500 kilo bajtów:

$ find /usr/bin -type f -size +500k -print

Aby wykonać na znalezionych plikach dowolną operację, należy skorzystać z opcji `exec`. Następujące polecenie usuwa wszystkie zrzuty `core` oraz kopie zapasowe `.bak` ze wszystkich katalogów lokalnego systemu plików:

$ find / -name 'core .bak' -xdev -exec rm

Poszukiwania rozpoczyna od katalogu (/). Wyszukuje wszystkie pliki o nazwach `core` lub kończące się na `.bak`. Każdy znaleziony plik jest kasowany za pomocą polecenia `rm`.

Polecenie `locate`

Polecenie to służy do szybkiego odnajdowania plików i katalogów w lokalnym systemie plików. Jest ono szybsze od `find` ponieważ zamiast przeczesywać twarde dyski korzysta z własnej bazy danych. Podobnie

jak `find, locate` dopuszcza stosowanie masek. Przykładowo aby znaleść wszystkie ikony edytora `emacs`, należy wpisać:

`$ locate iconemacs`

`locate` przeszukuje bazę danych znajdującą się w pliku `locatedb`, w katalogu find /var/lib. Do tworzenia owej bazy służy polecenie `updatedb`.

3 System pomocy

Wszystkie popularne dystrybucje Linuxa zawierają pełną dokumentację wiekszości swoich plików, poleceń i programów instalowanych na dysku twardym. Każda dystrybucja posiada szereg poleceń umożliwiających poznanie więcej informacji o systemie. Omówię teraz niektóre z nich.

3.1 Polecenie `man`

Polecenie `man` (ang. *manual*) wyświetla na ekranie wybrane fragmenty z elektronicznej dokumentacji Linuxa, noszącej nazwę stron podręcznika (ang. *manual pages*). Aby wyświetlić odpowiednią stronę, należy wpisać w wierszu poleceń `man` razem z interesującą nas nazwą pliku, programu itp. Jeśli chcemy zaczerpnąć informacji na temat systemu `man`, skorzystamy z następującej składni:

$ man man

Polecenie to wyświetli na ekranie stronę z podręcznika dotyczącą `man`, korzystając z programu `less`. Strony podręcznika są plikami tekstowymi, zapisanymi w specjalnym formacie. Aby dowiedzieć się więcej o tym formacie, można wyświetlić odpowiednią stronę:

`$ man 7 man`

Strona podręcznika i dodatkowa dokumentacja znajdują się w katalogu `/usr/man` i są podzielone na kilka działów, według typu i rodzaju opisywanych poleceń. Działy są zawarte w poniższej tabeli:

Dział	Typ dokumentacji
1	Polecenia (programy).
2	Polecenia systemowe (funkcje jądra).
3	Polecenia biblioteczne (funkcje języków programowania).
4	Pliki specjalne (pliki w katalogu `/dev`).
5	Formaty plików (`/etc/passwd` i innych).
6	Gry.
7	Pakiety makro informacji (format stron podręcznika itp.).
8	Administrowanie systemem (programy przeznaczone dla administratorów).
9	Procedury jądra (źródła).

3.2 Polecenie `whatis`

Polecenie `whatis` można używać, jeśli nie ma pewności co do funkcji danego programu. `whatis` wyświetla krótki opis działania wskazanego polecenia. Przykładowo napisanie:

`$ whatis cal`

spowoduje wyświetlenie następującej informacji:

```
cal (1)                  - displays a calendar
```

Opisy poszczególnych poleceń są odczytywane z odpowiednich stron podręcznika i umieszczane w bazie danych o nazwie `whatis`. Baza ta znajduje się w katalogu `/usr/man` i jest codziennie rekonstruowana.

3.3 Polecenie `apropos`

Polecenie `apropos` wykorzystuje bazę danych `whatis` do wyszukiwania wszystkich opisów zawierających podaną nazwę polecenia. Za jego pomocą można wyszukiwać programy pełniące funkcje podobne do innych znanych poleceń. Przykładowo, wpisanie:

`$ apropos bell`

przyniesie rezultat:

```
beep, flash (3) - Curses bell and screen flash routines
bell (n)        - Rings a display's bell
```

Jeśli w systemie nie zainstalowano polecenia `apropos` można zamiast niego używać `man` wraz z flagą `-K`. Wariant ten jest znacznie wolniejszy, ponieważ wymaga przeszukania całej zawartości podręcznika zamiast samych tylko opisów poleceń.

Gorąco zachęcam do czytania stron `man`, które są jednym z największych źródeł wiedzy o Linuxie. Można w nich znaleźć szczegółowe informacje dotyczące poleceń, w tym flagi znanych nam już poleceń, które nie zmieściły się na stronach tej pracy

4 Najważniejsze polecenia

Polecenie	**Opis**
`cd` ścieżka	Polecenie służące do zmiany katalogu bieżącego.
`cp` [flagi] *plik1 plik2*	Kopiuje *plik1* do *plik2*. Jeśli cel kopiowania istnieje, to plik te zostanie zastąpiony.
`cp` [flagi] *pliki katalog*	Kopiuje jeden lub więcej *plików* pod tą samą nazwą do *katalogu*, jeśli katalog docelowy istnieje, to pliki zostaną skopiowane do tego katalogu (*katalog* nie zostanie zastąpiony)
`find`	Niezwykle użyteczne polecenie do znajdywania pliku lub grupy plików specyficznego rodzaju. Użyteczną flagą jest `-xdev`, ogranicza ona obszar przeszukiwań do lokalnego systemu plików.
`ls` [flagi] *katalog*	Wyświetla listę zawartości katalogu . Jeśli nie zostanie podana ścieżka *katalog*, wyświetli zawartość bieżącego katlogu.
`man`[flagi][nazwa]	Wyświetla informacje zawarte w instrukcjach obsługi programów.
`mkdir` [flagi] *katalogi*	Tworzy jeden lub więcej *katalogów*.
`mv` [flagi] *źródło cel*	Przenosi pliki i katalogi lub zmienia ich nazwy.
`passwd`	Tworzy lub zmiennia hasło dla *użytkownika*.
`pwd`	Wyświetla pełną ścieżkę dostępu do obecnego katalogu roboczego.
`rm` [flagi] *pliki*	Usuwa jeden lub więcej *plików*.
`rm` -r *kataalog*	Polecenie `rm` z flagą -r usuwa *katalog* i jego zawartość wraz ze wszystkimi podkatalogami. UWAGA: użycie tej opcji może być niebezpieczne!
`rmdir`[flagi] *katalogi*	Usuwa *katalogi* o podanych nazwach, katalogi te muszą być puste.
`touch` [flagi] *pliki*	Służy do uaktualniania i tworzenia *plików*.