

3. Pliki w systemie Linux – interfejs użytkownika

Wstęp

Do najważniejszych zadań systemu operacyjnego należy wykonywanie programów i przechowywanie informacji dla użytkowników. Przechowywania wymagają zarówno kody wykonywalne samych programów, jak i dane, z których one korzystają. Wszystkie współczesne systemy przechowują informacje na fizycznych urządzeniach pamięci i udostępniają je pod postacią plików. System plików jest więc, obok zestawu programów systemowych, najbardziej widocznym składnikiem każdego systemu operacyjnego.

Wykład 3 zawiera opis systemu plików Linuxa z punktu widzenia użytkownika. Wprowadzamy tu pojęcie pliku i opisujemy niektóre jego atrybuty. Przedstawiamy organizację struktury katalogowej, omawiając przeznaczenie typowych katalogów oraz sposoby przeglądania i modyfikacji tej struktury. Podajemy zasady nazywania plików oraz ich ochrony. Prezentujemy również bogaty zestaw programów systemowych, umożliwiających manipulowanie plikami, przeglądanie i edycję ich wartości oraz archiwizację i kompresję plików.

3.1. Pliki i struktura katalogowa

Plik jest logiczną jednostką przechowywania informacji w pamięci pomocniczej. Zawiera ciąg bajtów, których znaczenie określa twórca pliku i korzystający z niego program.

System plików tworzy mechanizm bezpośredniego przechowywania i dostępu do informacji w systemie operacyjnym. Odzworowuje logiczną koncepcję pliku na fizyczne urządzenia pamięci.

Typy plików

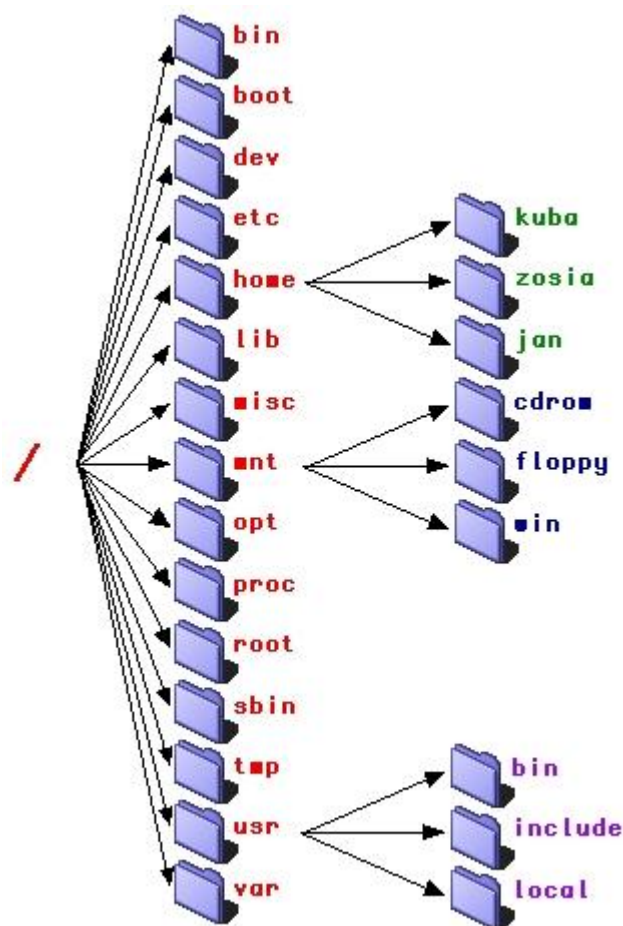
System Linux, podobnie jak system Unix, rozpoznaje 7 typów plików opisanych w tabelicy 3.1. Typ pliku przechowywany jest jako jeden z atrybutów pliku i nie ma żadnego odzwierciedlenia w jego nazwie.

Tablica 3.1 Typy plików w systemie Linux

Typ pliku	Przeznaczenie
Plik zwykły	Plik służący do przechowywania różnego rodzaju danych, np.: kodu wykonywalnego programu, kodu źródłowego, danych dla programu, tekstu.
Katalog	Plik przechowujący informacje o położeniu innych plików w strukturze katalogowej.
Plik specjalny blokowy	Plik reprezentujący urządzenie o dostępie blokowym.
Plik specjalny znakowy	Plik reprezentujący urządzenie o dostępie znakowym.
Dowiązanie symboliczne	Plik wskazujący na położenie innego pliku w strukturze katalogowej.
Kolejka FIFO	Plik implementujący łącze nazwane - jeden z mechanizmów komunikacji międzyprocesowej.
Gniazdo	Plik implementujący gniazdo w dziedzinie Unixa - jeden z mechanizmów komunikacji międzyprocesowej.

Struktura katalogowa

Pliki w systemie Linux zorganizowane są w drzewiastą strukturę katalogową z wyróżnionym korzeniem w postaci katalogu głównego */*. Typową strukturę katalogową przedstawiono na rys. 3.1 a przeznaczenie poszczególnych katalogów opisano w tabelicy 3.2.



Rys.3.1 Przykładowa struktura katalogowa w systemie Linux

Tablica 3.2 System plików - przeznaczenie katalogów

Nazwa katalogu	Zawartość
/	Katalog główny (ang. <i>root</i> - czyli korzeń drzewa katalogów).
/bin	Pliki z kodem wykonywalnym programów systemowych.
/boot	Pliki wykorzystywane w czasie uruchamiania systemu m.in. plik zawierający kod jądra systemu <i>vmlinuz</i> .
/dev	Pliki specjalne reprezentujące urządzenia wejścia/wyjścia.
/etc	Pliki konfiguracyjne jądra systemu oraz demonów realizujących usługi systemowe.
/lib	Pliki zawierające biblioteki funkcji.
/sbin	Pliki z kodem wykonywalnym programów służących do administracji systemem.
/tmp	Pliki tymczasowe.
/usr	Katalog o podobnej do katalogu głównego strukturze podkatalogów.
/var	Pliki konfiguracyjne i informacyjne systemu, m.in. skrzynki pocztowe użytkowników,

	pliki komunikatów systemowych.
/proc	Interfejs do struktur danych jądra opisujących wszystkie aktywne procesy.
/home	Katalogi domowe użytkowników.
/mnt	Typowe punkty dołączenia innych fragmentów systemu plików do wspólnej struktury katalogowej.
/media/cdrom	Punkt dołączenia dysku CD-ROM.
/media/floppy	Punkt dołączenia dyskietki.
/media/disk	Punkt dołączenia nośnika USB.
/opt	Opcjonalne, dodatkowe oprogramowanie.

W każdym katalogu istnieją dwie specjalne pozycje: `.` i `..` wskazujące odpowiednio na katalog bieżący i katalog nadrzędny.

Nazwa każdego pliku (również katalogu) może składać się z 255 znaków. System rozróżnia małe i duże litery oraz dopuszcza stosowanie dowolnych znaków specjalnych. Użycie niektórych znaków, takich jak spacja czy tabulator, wymusza jednak specjalne traktowanie nazw plików ze względu na możliwość niewłaściwej interpretacji przez program interpretera poleceń. Nazwy takie podaje się zazwyczaj w cudzysłowach lub zabezpiecza w inny sposób (patrz Wykład 5).

Pliki, których nazwy rozpoczynają się od kropki, określane są jako pliki ukryte i traktowane w specjalny sposób przez niektóre polecenia (patrz opis polecenia `ls`). Nazwy plików nie są unikalne, więc w różnych katalogach mogą występować pliki o tej samej nazwie.

Plik można w pełni zidentyfikować podając jego nazwę ścieżkową, składającą się z właściwej nazwy pliku oraz ścieżki dostępu w strukturze katalogowej. Ścieżka zawiera nazwy kolejnych katalogów, prowadzących do pliku, rozdzielone znakiem `/`.

Ścieżka bezwzględna rozpoczyna się od katalogu głównego `/` np.: `/usr/local/bin`.
Ścieżka względna ustalana jest zawsze względem katalogu bieżącego i rozpoczyna się od nazwy jednego z jego podkatalogów (brak znaku `/` na początku) np.: `local/bin` , `./local/bin` , `../bin`.

W nazwach ścieżkowych można również posługiwać się znakiem specjalnym `~` oznaczającym katalog domowy zalogowanego użytkownika np.: `~/Mail` . Zapis `~użytkownik` oznacza natomiast katalog domowy wskazanego użytkownika.

Pliki nie rezydują fizycznie w katalogach. Katalog zawiera jedynie listę pozycji wiążących nazwy plików z ich fizycznym położeniem w systemie plików.

W ten sam sposób dowiązane są pliki wszystkich typów, a więc również podkatalogi do katalogu nadrzędnego. Taki zapis w katalogu nosi nazwę **dowiązania** albo **dowiązania twardego** (ang. *hard link*). Możliwość utworzenia wielu dowiązań oznacza, że ten sam plik może być widoczny pod różnymi nazwami w kilku katalogach. Nie dotyczy to jednak podkatalogów. Zakaz tworzenia dowiązań twardych do istniejących katalogów zapobiega powstawaniu cykli w strukturze katalogowej.

Istnieje również drugi rodzaj dowiązań - **dowiązanie symboliczne** (ang. *symbolic link*). Jest to plik specjalnego typu, który wskazuje na położenie innego pliku w strukturze katalogowej, czyli przechowuje jego pełną nazwę ścieżkową. Dowiązanie symboliczne może wskazywać na dowolny plik lub katalog. Co więcej, może nawet wskazywać na nazwę, która nie istnieje.

Przeglądanie struktury katalogowej umożliwiają trzy podstawowe polecenia: **`pwd`**, **`cd`** i **`ls`**.

Polecenie **`pwd`** wypisuje pełną nazwę ścieżkową katalogu bieżącego. Zmianę katalogu bieżącego, czyli przejście do innego katalogu, umożliwia polecenie **`cd`**:

cd [katalog]

W przypadku braku argumentu **katalog**, domyślnie następuje przejście do **katalogu domowego** użytkownika.

Program **ls** wyświetla informacje o plikach podanych jako argumenty wywołania lub o plikach zawartych w podanych katalogach:

ls [opcje] [plik...]

gdzie:

plik - nazwa pliku (również katalogu),

opcje - przełączniki określające rodzaj i format wyświetlanych informacji.

Program wywołany bez argumentów wyświetla zawartość katalogu bieżącego. Domyślnie nie są wyświetlane informacje o plikach, których nazwy zaczynają się od znaku "." (kropka), czyli mają postać **.nazwa** np. **. .** lub **.bashrc**. Takie pliki nazywane są plikami ukrytymi. Zakres informacji o plikach podawanych przez program **ls** można modyfikować za pomocą bogatego zestawu opcji. Najważniejsze z nich zamieszczamy poniżej:

-a - wyświetla wszystkie plik, również ukryte,

-A - nie wyświetla pozycji **.** i **..**,

-d - nie wyświetla zawartości katalogów, podanych jako argumenty, jedynie ich nazwy,

-i - wyświetla dodatkowo numery i-węzłów plików,

-l - wyświetla więcej atrybutów plików (długi format).

Polecenie **ls -l** jest jednym z najczęściej wykorzystywanych. Wyświetla ono w kolejnych kolumnach następujące atrybuty plików:

1. typ i prawa dostępu w postaci symbolicznej,
2. liczbę dowiezań,
3. nazwę właściciela i grupy,
4. rozmiar,
5. datę ostatniej modyfikacji,
6. nazwę pliku.

Pierwsza kolumna składająca się z 10-ciu znaków przedstawia symbolicznie typ pliku (1 znak) i prawa dostępu (9 znaków). Symbole określające typ plik to:

- - plik zwykły,

d - katalog,

b - plik specjalny blokowy,

c - plik specjalny znakowy,

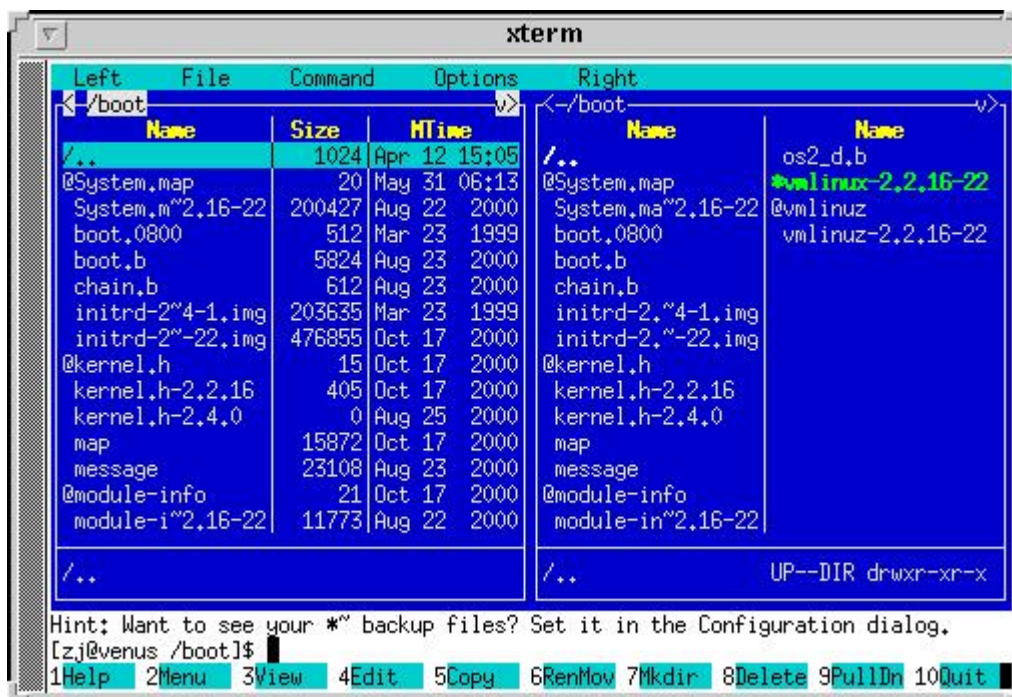
l - dowiezanie symboliczne,

p - FIFO,

s - gniazdo.

Znaczenie praw dostępu zostało opisane w punkcie 3.2.

Jeśli użytkownik korzysta z sesji graficznej to do przeglądania struktury katalogowej można wykorzystać liczne programy menedżerów plików. Pomocny może być również program o nazwie Midnight Commander, którego interfejs, pokazany na rys. 3.2 jest podobny do programów Norton Commander i Windows Commander. Program umożliwia oglądanie zawartości katalogów, manipulację plikami, przeglądanie i edycję zawartości plików. Wersję programu działającą w środowisku tekstowym uruchamiamy poleceniem **mc**, zaś wersję działającą w środowisku graficznym - poleceniem **gmc**.



Rys.3.2 Zastosowanie programu Midnight Commander do przeglądania plików

3.2. Ochrona plików

Systemy przeznaczone dla wielu użytkowników muszą zapewnić każdemu użytkownikowi ochronę jego zasobów, a w szczególności ochronę jego plików. W systemach Linux i Unix ochrona plików realizowana jest poprzez **prawa własności** i **prawa dostępu** do plików. Informacje te przechowywane są w systemie plików w postaci trzech atrybutów każdego pliku.

Prawa własności

Prawa własności opisane są przez dwa atrybuty pliku:

1. **identyfikator użytkownika** będącego właścicielem pliku,
2. **identyfikator grupy** użytkowników posiadających specjalne uprawnienia do pliku.

Atrybuty te ustawiane są zgodnie z wartościami UID i GID użytkownika tworzącego plik. Wartości te mogą być później zmienione przy pomocy następujących poleceń:

```
chown [opcje] właściciel[:grupa] plik ...
```

```
chown [opcje] :grupa plik ...
```

```
chgrp [opcje] grupa plik ...
```

gdzie:

właściciel - nazwa nowego właściciela pliku,

grupa - nazwa nowego właściciela pliku,

plik - nazwa nowego właściciela pliku.

Polecenie **chown** umożliwia zmianę właściciela pliku i/lub grupy użytkowników w zależności od sposobu wywołania. Polecenie **chgrp** zmienia grupę użytkowników wskazanym plikom. Dowolnych zmian tych atrybutów dla każdego pliku może dokonać wyłącznie administrator (użytkownik **root**). Zwykły użytkownik może jedynie ustawić swoim plikom numer jednej z grup, do których sam należy.

Programy **chown** i **chgrp** wykorzystują ten sam zestaw opcji:

-R - rekurencyjnie zmienia atrybuty plików

-f - nie wyświetla większości informacji o błędach

-v - wyświetla diagnostykę plików o zmienionych atrybutach

Prawa dostępu

Kontrolowany dostęp do plików może być realizowany na wiele sposobów. Podstawowym mechanizmem stosowanym w systemach Linux i UNIX są tzw. **dostępy grupowe**. Oznacza to, że prawa dostępu do plików nie są nadawane indywidualnym użytkownikom, ale pewnym kategoriom (grupom) użytkowników. Wyróżnione zostały trzy takie kategorie:

1. właściciel pliku,
2. grupa użytkowników, określona przez prawa własności,
3. pozostali użytkownicy.

Dla każdej z powyższych kategorii zdefiniowane są trzy rodzaje uprawnień:

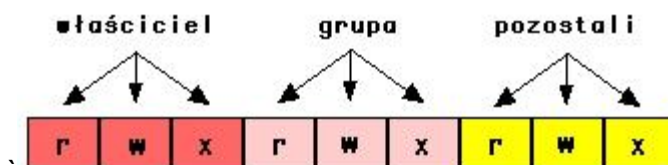
1. prawo czytania z pliku - oznaczone literą **r**,
2. prawo pisania do pliku - oznaczone literą **w**,
3. prawo wykonywania pliku - oznaczone literą **x**.

Znaczenie poszczególnych praw w odniesieniu do plików zwykłych i katalogów opisano w Tablicy 3.3.

Tablica 3.3 Znaczenie praw dostępu do pliku zwykłego i katalogu

	Plik zwykły	Katalog
Prawo czytania	Prawo odczytania zawartości pliku, prawo kopiowania pliku.	Prawo odczytania zawartości katalogu, czyli odczytania listy plików poleceniem ls.
Prawo pisania	Prawo modyfikacji zawartości pliku poprzez dopisanie, nadpisanie lub usunięcie fragmentów.	Prawo modyfikacji zawartości katalogu, czyli dodania bądź usunięcia pliku z katalogu.
Prawo wykonywania	Prawo uruchomienia programu lub skryptu zapisanego w pliku.	Prawo przejścia do katalogu poleceniem cd.

Wszystkie prawa dostępu przechowywane są łącznie w jednym atrybucie pliku jako 9 kolejnych bitów, przy czym odpowiedni bit jest ustawiony na '1', jeśli prawo jest przyznane. Symbolicznie prawa dostępu przedstawiane są w postaci 9 pól wypełnionych literami r, w, x lub kreskami, co widać na rysunku 3.3. Litera oznacza nadanie odpowiedniego prawa, podczas gdy kreska - brak danego prawa.



Rys. 3.3 Prawa dostępu do pliku

Prawa dostępu mogą być modyfikowane przez właściciela pliku lub administratora. Właściciel sam ustala, jakie prawa do swoich plików przyzna poszczególnym kategoriom użytkowników. Również sobie musi ustawić odpowiednie prawa dostępu, aby korzystać z własnych plików. Właściwie dobrane prawa dostępu mogą posłużyć jako zabezpieczenie przed przypadkowym zniszczeniem zawartości wybranych plików przez właściciela. Polecenie **chmod** pozwala dokonać wszystkich zmian:

```
chmod [opcje] tryb[, tryb]... plik ...
```

```
chmod [opcje] tryb_liczbowy plik ...
```

gdzie:

tryb - prawa dostępu w postaci symbolicznej,
tryb_liczbowy - prawa dostępu w kodzie ósemkowym,
plik - nazwa pliku.

Nowe uprawnienia mogą być podane w postaci symbolicznej lub numerycznej. Stosując postać symboliczną, należy wyspecyfikować w postaci argumentu **tryb** jakie uprawnienia, dla kogo i w jaki sposób modyfikujemy. Argument tryb składa się z trzech pól **kategoriaoperatorprawa**. Znaczenie poszczególnych pól jest wyjaśnione w tablicy 3.4:

Tablica 3.4 Znaczenie pól w trybie pliku

Pole	Znaczenie	Zawartość
kategoria	kategoria użytkowników	dowolna kombinacja znaków: <ul style="list-style-type: none"> • u (właściciel) • g (grupa) • o (pozostali użytkownicy) • a (wszyscy)
operator	rodzaj operacji na prawach dostępu	jeden ze znaków: <ul style="list-style-type: none"> • + (dodanie) • - (odebranie) • = (ustawienie wyłącznie uprawnień podanych w poleceniu)
prawa	prawa dostępu	dowolna kombinacja znaków: <ul style="list-style-type: none"> • r • w • x

Elementy **kategoria** i **prawa** mogą zostać pominięte w argumencie **tryb** polecenia **chmod**. Brak kategorii oznacza, że operacja zmiany praw dotyczy wszystkich użytkowników.

Przykład

```
chmod u+wx plik      - dodanie prawa czytania i pisania dla właściciela pliku,
chmod +x plik        - dodanie prawa wykonywania dla wszystkich użytkowników,
chmod go-w plik      - odebranie prawa pisania dla grupy i innych użytkowników.
```

W trybie numerycznym prawa dostępu opisuje się w postaci trzech cyfr w kodzie ósemkowym. Każda cyfra koduje wartości trzech bitów praw dostępu, czyli zapisuje uprawnienia jednej kategorii użytkowników.

Przykład

```
chmod 755 plik      - ustawia następujące prawa: rwx r-x r-x,
chmod 640 plik      - ustawia następujące prawa: rw- r-- ---.
```

Początkowe ustawienia praw dostępu dla tworzonych plików określone są na podstawie parametru **umask** jako dopełnienie jego wartości do pełnych uprawnień, czyli **tryb = 777 - umask**.

Maska uprawnień do tworzonych plików powinna być zdefiniowana w sesji każdego użytkownika. Służy do tego polecenie **umask** o następującej składni:

`umask [-S] maska`

Argument **maska** może być podany w postaci numerycznej lub w postaci symbolicznej. Postać numeryczna opisuje w kodzie ósemkowym dopełnienie praw dostępu, czyli te prawa, których nie nadajemy. W postaci symbolicznej stosujemy składnię jak w poleceniu **chmod**. Polecenie wywołane bez argumentu wypisuje aktualną maskę praw dostępu w postaci numerycznej lub w postaci symbolicznej jeśli wywołane zostanie z opcją **-S**.

3.3. Manipulowanie plikami

Podstawowe operacje manipulowania plikami obejmują tworzenie i usuwanie plików, zmiany nazw, przesuwanie pomiędzy katalogami, kopiowanie oraz tworzenie dowiązań.

Nowy plik może powstać w systemie w wyniku działania wielu różnych programów. Utworzenie nowego katalogu wymaga zazwyczaj użycia specjalnego polecenia:

```
mkdir [opcje] katalog ...
```

Pusty katalog (zawierający jedynie pozycje . i ..) można usunąć poleceniem:

```
rmdir katalog ...
```

Jeżeli katalog jest niepusty, trzeba użyć polecenia, które usuwa całą gałąź drzewa rozpoczynającą się od wskazanego katalogu:

```
rm -r|R katalog ...
```

Polecenie **rm** w ogólnej postaci umożliwia usunięcie dowolnego pliku:

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

Użytkownik musi posiadać jednak odpowiednie uprawnienia do katalogu, w którym znajduje się plik. Same uprawnienia do pliku nie mają tu znaczenia, gdyż operacja polega na usunięciu zapisu z katalogu.

Najważniejsze opcje polecenia **rm** opisano poniżej:

- f - tryb forsowny, usuwa istniejące pliki docelowe bez prośby o potwierdzenie,
- i - tryb interaktywny, prosi o potwierdzenie operacji dla każdego argumentu,
- r - rekurencyjnie usuwa zawartość katalogów,
- R - rekurencyjnie usuwa zawartość katalogów.

Zmiana nazwy pliku i przesuwanie pliku pomiędzy katalogami, to w istocie te same operacje polegające na usunięciu jednego wpisu w katalogu i dodaniu innego. Polecenie **mv** umożliwia wykonanie obydwu operacji, Jego składnia jest następująca:

```
mv [opcje] plik_źródłowy plik_docelowy
```

```
mv [opcje] plik... katalog
```

gdzie:

plik_źródłowy - nazwa pliku,

plik_docelowy - nazwa pliku,

plik - nazwa pliku dowolnego typu (również katalogu),

katalog - nazwa katalogu docelowego.

Jeżeli plik docelowy istnieje, to zostanie usunięty pod warunkiem posiadania odpowiednich uprawnień do katalogu. Użycie jednej z poniższych opcji wpływa na przebieg tej operacji:

- f - tryb forsowny, usuwa istniejące pliki docelowe bez prośby o potwierdzenie,
- i - tryb interaktywny, prosi o potwierdzenie operacji,

Do kopiowania plików służy polecenie **cp** o składni:

```
cp [opcje] plik_źródłowy plik_docelowy  
cp [opcje] plik... katalog
```

gdzie:

plik_źródłowy - nazwa pliku,

plik_docelowy - nazwa pliku,

plik - nazwa pliku dowolnego typu (również katalogu),

katalog - nazwa katalogu docelowego.

Możemy skopiować jeden plik na drugi lub kilka plików do katalogu. Poprzez dobór odpowiednich opcji można określić sposób wykonania tej operacji np. wymusić rekurencyjne kopiowanie całej gałęzi drzewa katalogów. Najważniejsze opcje przedstawiono poniżej:

- f - tryb forsowny, usuwa istniejące pliki docelowe bez prośby o potwierdzenie,
- i - tryb interaktywny, prosi o potwierdzenie operacji,
- r - rekurencyjnie kopiuje katalogi,
- R - rekurencyjnie kopiuje katalogi,
- p - zachowuje atrybuty pliku oryginalnego.

Polecenie **ln** umożliwia stworzenie nowego dowiązania twardego lub dowiązania symbolicznego:

```
ln [-s] plik_źródłowy plik_docelowy
```

gdzie:

plik_źródłowy - nazwa istniejącego pliku (dowiązania),

plik_docelowy - nazwa nowego dowiązania,

-s - tworzenie dowiązania symbolicznego.

3.4. Przeglądanie i edycja plików

Edycja i przeglądanie zawartości plików w systemie Linux nie różni się w szczególny sposób od innych systemów, takich jak MS Windows czy Mac OS. W tym segmencie wykładu przedstawione zostanie, na zasadzie ogólnej informacji, najczęściej spotykane oprogramowanie.

Przeglądanie zawartości pliku

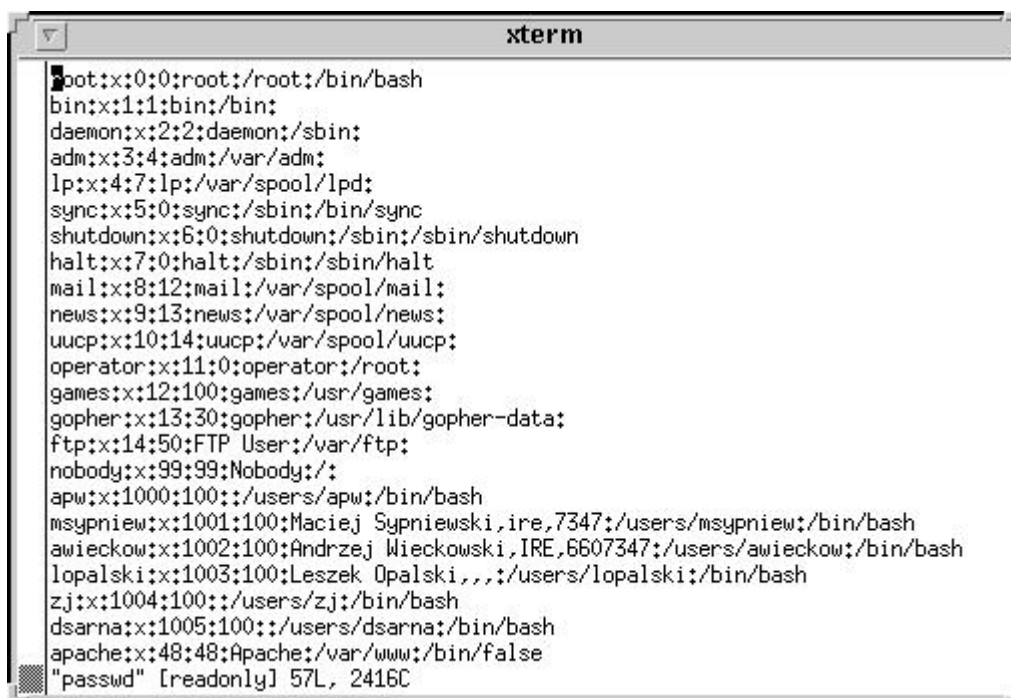
Przeglądanie pliku to nic innego jak wyświetlenie jego zawartości na ekranie i operacja taka ma sens oczywiście tylko w przypadku plików tekstowych. W tym celu można użyć takich programów jak **more**, **page**, **less** czy **cat**. Pierwsze trzy z nich charakteryzują się tym, że treść pliku prezentowana jest strona po stronie. Z tego powodu są one wykorzystywane przez system pomocy systemowej **man** do wyświetlania odpowiednio sformatowanych dokumentów opisujących działanie i sposób wykorzystania licznych w systemie Linux programów użytkowych (więcej informacji można uzyskać wydając polecenie **man man**). Ponadto, wspomniane programy są bardzo często używane do tworzenia bardziej złożonych poleceń z wykorzystaniem przetwarzania potokowego, np.:

```
ps -ef | more
more passwd | grep zj
```

Warto zwrócić tu uwagę na polecenie **cat**. Kopiuje ono dane wejściowe na standardowy strumień wyjściowy. Zastosowania polecenia **cat** mogą być bardziej "twórcze" niż samo wyświetlenie jego zawartości. Może to być scalenie zawartości kilku plików bez uciekania się do pomocy edytora jak na pokazanym poniżej przykładzie:

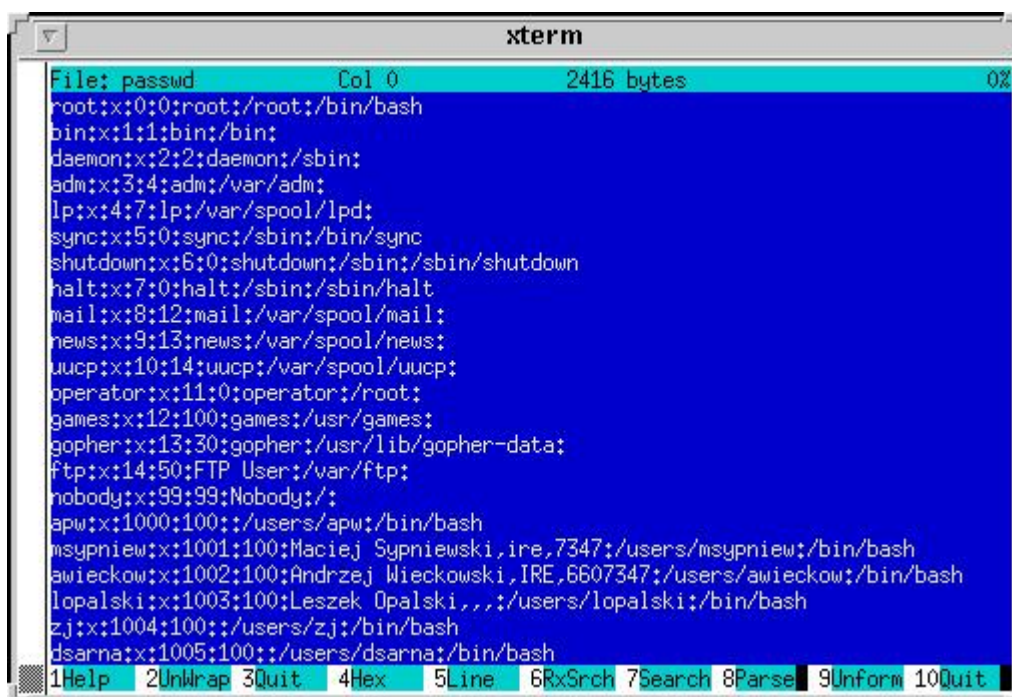
```
cat plik1 pliki2 plik3 > plik_wynikowy
```

Do przeglądania zawartości plików można też wykorzystać edytory w trybie pracy "tylko do czytania" (ang. *read only*). Warto zauważyć, że taki tryb pracy edytora jest włączany automatycznie, gdy użytkownik otwiera do edycji plik, do którego nie posiada uprawnień do pisania (prawo **w**). Przykład takiej operacji, wywołanie edytora **vi** do edycji pliku **passwd**, pokazano na rys. 3.4.



Rys. 3.4 Edytor vi - próba edycji passwd pliku przy braku uprawnień do pisania

Podobny efekt osiągniemy próbując wykorzystać do tego celu edytor wbudowany do programu Midnight Commander (rys 3.5).



Rys. 3.5 Midnight Commander - próba edycji pliku passwd przy braku uprawnień do pisania

Edytory

Jak już powiedziano, edycja plików w systemie Linux nie wyróżnia się niczym szczególnym w porównaniu z innymi systemami operacyjnymi. Dostępne edytory można podzielić na dwie grupy: do pracy w trybie tekstowym i trybie graficznym. Do pierwszej z nich należą m.in. **vi**, **vim**, **pico**, **joe**, **emacs** czy też edytor wbudowany do programu Midnight Commander. Drugą grupę stanowią programy **gvim** (wersja edytora **vim**), **Nedit**, **Xemacs** (wersja edytora **emacs** dla środowiska X Window) czy narzędzia dostępne w środowiskach systemu aktywnego pulpitu (np. KDE lub GNOME). Oprogramowanie to jest zwykle dostępne bezpłatnie jako *freeware* lub na warunkach licencji GPL (GNU General Public License).

Oprócz edytorów istnieje też grupa procesorów tekstu dostępnych na platformę Linux. Warto tu zwrócić uwagę na narzędzia dostępne w pakietach **Open Office** i **Libre Office**. Istnieje też oprogramowanie komercyjne, które nie będzie tu omawiane.

*Autorzy tego wykładu zalecają, aby każdy użytkownik zapoznał się z co najmniej jednym edytorem działającym w trybie tekstowym. Jest to istotne z tego względu, iż stosunkowo często występuje konieczność pracy właśnie w trybie tekstowym a nie graficznym. Przykłady takich sytuacji to np. praca zdalna z wykorzystaniem wolnego połączenia (np. modemowego), zdalne korzystanie z poczty elektronicznej za pomocą protokołu telnet, ssh, itp. Wszystkich, którzy zamierzają stać się administratorami namawiamy do opanowania edytora **vi**. Jest to jedyny edytor uznany za w pełni stabilny i w związku z tym bezpieczny w użytkowaniu, co jest szczególnie ważne dla administratora. Ponadto **vi** jest dostępny w każdym systemie unixowym. Warto tu zaznaczyć, że mimo istnienia szeregu znacznie wygodniejszych w użytkowaniu narzędzi posłużenie się edytorem **vi** może okazać się jedyną szansą na poprawienie błędów w plikach konfiguracyjnych, które te błędy uniemożliwiają uruchomienie sesji graficznej.*

Edytory pracujące w trybie tekstowym - edytor vi

W dalszym ciągu tej lekcji będzie zaprezentowana krótka instrukcja pracy z edytorem **vi** oraz będą zaprezentowane inne typowe edytory.

Specyfika pracy z edytorem vi polega na tym, że edytor ten ma dwa oddzielne tryby pracy:

- tryb wydawania poleceń,
- tryb wprowadzania tekstu.

W trybie wydawania poleceń wprowadzone za pomocą klawiatury znaki są interpretowane jako polecenia (np. przemieszczania kursora, kopiowania i kasowania tekstu, zapisu do pliku, wyszukiwania i/lub zamiany wzorców).

W trybie wprowadzania tekstu wszystkie znaki są traktowane jako treść pliku i nie ma możliwości wydania żadnego polecenia.

Użytkownik musi więc w trakcie pracy wielokrotnie przełączać się z jednego trybu w drugi, co obok dużej liczby poleceń stanowi istotną niedogodność - zwłaszcza dla początkujących. W miarę nabywania wprawy w posługiwanie się edytorem **vi** niedogodności te przestają przeszkadzać i ustępują miejsca zaletom - np. szerokim możliwościom niedostępnym w innych edytorach.

Dla wszystkich, którzy nie byli w stanie zaakceptować "surowej elegancji" edytora **vi** opracowano jego "ucywilizowaną" wersję - **vim**. Trzeba jednak pamiętać, że **vim** ma stosunkowo krótką historię w porównaniu z **vi** i nie można go obdarzać równie dużym zaufaniem.

Po uruchomieniu edytora **vi** znajduje się on w trybie wydawania poleceń. W celu przejścia do trybu wprowadzania tekstu należy wydać jedno z poleceń: **i**, **I**, **a**, **A**, **o**, **O**, a więc wykonać operację rozpoczęcia wstawiania lub dopisywania w bieżącym (lub sąsiednim) wierszu. Jeśli w trakcie wprowadzania tekstu użytkownik zamierza wydać polecenie (np. zapisu do pliku i zakończenia pracy) musi najpierw zakończyć tryb wprowadzania tekstu przez naciśnięcie klawisza **ESC**. Powrót do trybu wprowadzania wymaga ponownego wydania jednego z poleceń: **i**, **I**, **a**, **A**, **o**, **O**. W tablicy 3.5 zebrano najbardziej podstawowe polecenia edytora **vi**.

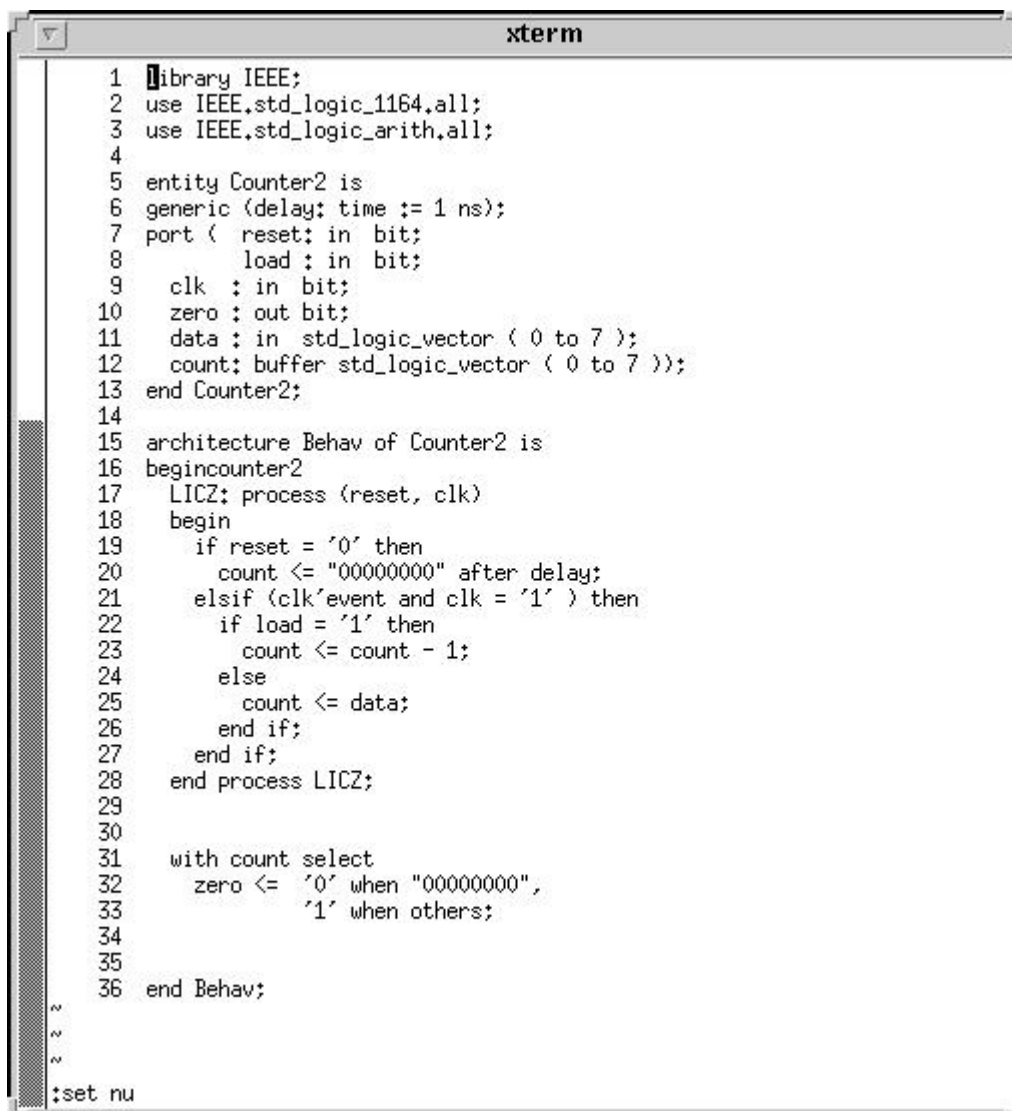
Tablica 3.5 Wybrane polecenia edytora vi

Zapisywanie tekstu i kończenie pracy z edytorem:	
:w	zapis do bieżącego pliku
:w nazwa	nazwa zapis do pliku o podanej nazwie
:w! nazwa	zapis do pliku o podanej nazwie z zamazaniem poprzedniej zawartości
:x,y w nazwa	zapis wierszy od numeru x do y do pliku o podanej nazwie
:x,y w! nazwa	zapis wierszy od numeru x do y do pliku o podanej nazwie z zamazaniem poprzedniej zawartości
:wq	zapis bieżącego pliku i wyjście z edytora
ZZ	zapis bieżącego pliku i wyjście z edytora
:q	wyjście z edytora
:q!	wyjście z edytora bez zapisywania dokonanych zmian
Przemieszczanie kursora:	
←↑→↓	zgodnie z kierunkiem strzałki

\$	na koniec wiersza
0	na początek wiersza
^	do pierwszego znaku w wierszu, nie będącego spacją ani znakiem tabulacji
G	do ostatniego wiersza w pliku
nG	do n-tego wiersza w pliku
Wprowadzanie tekstu:	
i	wstawianie tekstu przed kursorem
I	wstawianie tekstu na początku wiersza
a	dodawanie tekstu za kursorem
A	dodawanie tekstu na końcu wiersza
o	utworzenie pustego wiersza za wierszem bieżącym i przejście do trybu edycji
O	utworzenie pustego wiersza przed wierszem bieżącym i przejście do trybu edycji
Usuwanie tekstu:	
x	znaku znajdującego się pod kursorem
X	znaku znajdującego się przed kursorem
nx	n-znaków zaczynając od bieżącej pozycji kursora
dkomenda_kursora	usuwanie tekstu od bieżącej pozycji kursora do pozycji określonej przez polecenie <i>komenda_kursora</i>
dd	bieżącego wiersza
ndd	kolejnych n-wierszy poczynając od bieżącego
D	od pozycji kursora do końca bieżącego wiersza
d0	od pozycji kursora do początku bieżącego wiersza
dG	wszystkich wierszy do końca pliku, poczynając od bieżącego wiersza
p	wstawienie usuniętego tekstu za kursorem
P	wstawienie usuniętego tekstu przed kursorem
xp	zamiana pozycji dwóch sąsiednich znaków
Wycofywanie zmian:	
u	powrót do stanu sprzed ostatniego polecenia
U	wczytanie bieżącego wiersza z pliku do bufora

:q!	wyjście z edytora bez zapisywania dokonanych zmian
Modyfikowanie tekstu:	
r znak	zamiana bieżącego znaku
s znaki ESC	zamiana bieżącego znaku na ciąg znaków
cwtekst ESC	zamiana bieżącego wyrazu na ciąg znaków
Przewijanie:	
CTRL-U	do tyłu o pół ekranu
CTRL-D	do przodu o pół ekranu
CTRL-B	do tyłu o cały ekran
CTRL-F	do przodu o cały ekran
Kopiowanie i wstawianie:	
yy	kopiowanie do bufora bieżącej wiersza
nyy	kopiowanie do bufora n-kolejnych wierszy
p	wstawianie tekstu z bufora za kursorem
P	wstawianie tekstu z bufora przed kursorem
.	powtarzanie ostatnio wydanego polecenia
Wyszukiwanie wzorców:	
/wzorzec	poszukiwanie wzorca w przód od pozycji kursora
?wzorzec	poszukiwanie wzorca w tył od pozycji kursora
n	kontynuacja poszukiwania w tym samym kierunku
N	kontynuacja poszukiwania w przeciwnym kierunku
Wyszukiwanie i zamiana wzorców	
: zakres_wierszy s/wzorzec_do_znalezienia/nowy_tekst/	
: zakres_wierszy s/wzorzec_do_znalezienia/nowy_tekst/g	
: zakres_wierszy s/wzorzec_do_znalezienia/nowy_tekst/cg	
gdzie zakres_wierszy	
%	oznacza wszystkie wiersze w pliku
nr_w1,nr_w2	oznacza wszystkie wiersze o numerach ze wskazanego zakresu

Na rysunkach 3.6 - 3.10 pokazano efekty wykonywania niektórych poleceń. I tak na rys.3.6 widać rezultat wykonania polecenia `:set nu`, które włącza opcję numerowania wierszy tekstu. Konstrukcja `:set opcja` umożliwia włączanie szeregu innych opcji zmieniających standardowe zachowanie edytora.



```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4
5 entity Counter2 is
6 generic (delay: time := 1 ns);
7 port ( reset: in bit;
8       load : in bit;
9       clk  : in bit;
10      zero : out bit;
11      data : in std_logic_vector ( 0 to 7 );
12      count: buffer std_logic_vector ( 0 to 7 ));
13 end Counter2;
14
15 architecture Behav of Counter2 is
16 begin
17   LICZ: process (reset, clk)
18   begin
19     if reset = '0' then
20       count <= "00000000" after delay;
21     elsif (clk'event and clk = '1' ) then
22       if load = '1' then
23         count <= count - 1;
24       else
25         count <= data;
26       end if;
27     end if;
28   end process LICZ;
29
30   with count select
31     zero <= '0' when "00000000",
32            '1' when others;
33
34
35
36 end Behav;
~
~
~
:set nu

```

Rys. 3.6 Edytor vi - efekt wykonania polecenia `:set nu`

Na rysunkach 3.7 i 3.8 pokazano działanie polecenia zamiany wzorców `:%s/count/stan/`. Zgodnie z informacjami podanymi w tablicy 3.5 zamiany dokonano w całym tekście (na co wskazuje znak `%`). W rezultacie wykonano 6 zmian słowa **count** na **stan**. Dociekliwi zauważą zapewne, że w wierszu numer 23 słowo **count** występowało dwukrotnie a zamianie poddano tylko pierwsze wystąpienie. Jest to poprawne zachowanie edytora. Jeśli użytkownik życzy sobie, by zamianie podlegały wszystkie wystąpienia wzorca w każdym wierszu, powinien dodać do polecenia modyfikator **g**, który powoduje "globalne" wykonywanie podstawienia. Polecenie takie - `:%s/count/stan/g` - oraz efekty jego działania pokazano na rysunkach 3.9 i 3.10.

```
xterm
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4
5 entity Counter2 is
6 generic (delay: time := 1 ns);
7 port ( reset: in bit;
8       load : in bit;
9       clk  : in bit;
10      zero : out bit;
11      data : in std_logic_vector ( 0 to 7 );
12      count: buffer std_logic_vector ( 0 to 7 ));
13 end Counter2;
14
15 architecture Behav of Counter2 is
16 begincounter2
17   LICZ: process (reset, clk)
18   begin
19     if reset = '0' then
20       count <= "00000000" after delay;
21     elsif (clk'event and clk = '1' ) then
22       if load = '1' then
23         count <= count - 1;
24       else
25         count <= data;
26       end if;
27     end if;
28   end process LICZ;
29
30
31   with count select
32     zero <= '0' when "00000000",
33           '1' when others;
34
35
36 end Behav;
```

~
~
~

```
:%s/count/stan/
```

Rys. 3.7 Edytor vi - stan przed wydaniem polecenia :%s/count/stan/

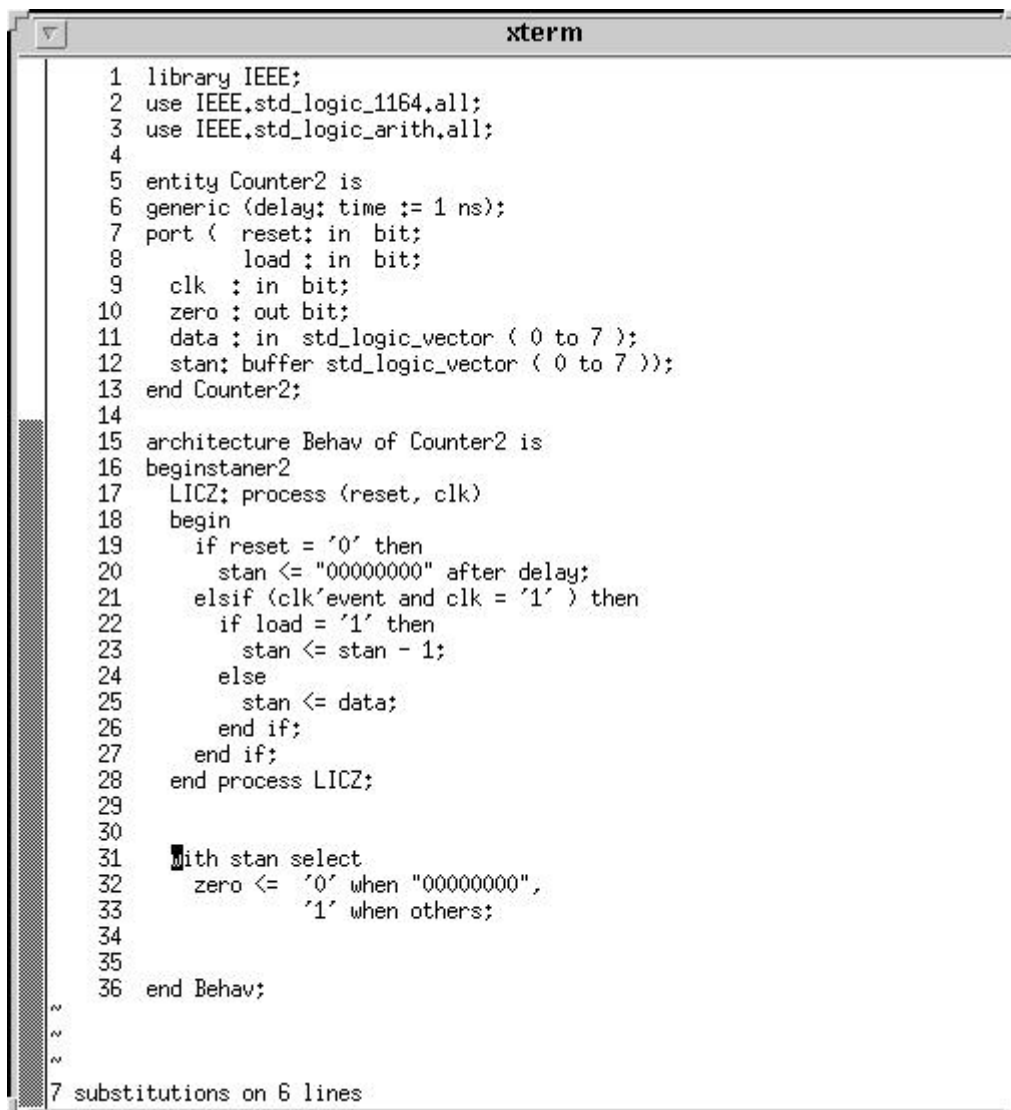
```
xterm
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4
5 entity Counter2 is
6 generic (delay: time := 1 ns);
7 port ( reset: in bit;
8       load : in bit;
9       clk  : in bit;
10      zero : out bit;
11      data : in std_logic_vector ( 0 to 7 );
12      stan: buffer std_logic_vector ( 0 to 7 ));
13 end Counter2;
14
15 architecture Behav of Counter2 is
16 begin
17   LICZ: process (reset, clk)
18   begin
19     if reset = '0' then
20       stan <= "00000000" after delay;
21     elsif (clk'event and clk = '1' ) then
22       if load = '1' then
23         stan <= count - 1;
24       else
25         stan <= data;
26       end if;
27     end if;
28   end process LICZ;
29
30   With stan select
31     zero <= '0' when "00000000",
32            '1' when others;
33
34
35
36 end Behav;
```

6 substitutions

Rys. 3.8 Edytor vi - stan po wykonaniu polecenia :%s/count/stan/

```
xterm
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4
5 entity Counter2 is
6 generic (delay: time := 1 ns);
7 port ( reset: in bit;
8       load : in bit;
9       clk  : in bit;
10      zero : out bit;
11      data : in std_logic_vector ( 0 to 7 );
12      count: buffer std_logic_vector ( 0 to 7 ));
13 end Counter2;
14
15 architecture Behav of Counter2 is
16 begincounter2
17   LICZ: process (reset, clk)
18   begin
19     if reset = '0' then
20       count <= "00000000" after delay;
21     elsif (clk'event and clk = '1' ) then
22       if load = '1' then
23         count <= count - 1;
24       else
25         count <= data;
26       end if;
27     end if;
28   end process LICZ;
29
30   with count select
31     zero <= '0' when "00000000",
32           '1' when others;
33
34
35
36 end Behav;
~
~
~
:%s/count/stan/g
```

Rys. 3.9 Edytor vi - stan przed wydaniem polecenia :%s/count/stan/g

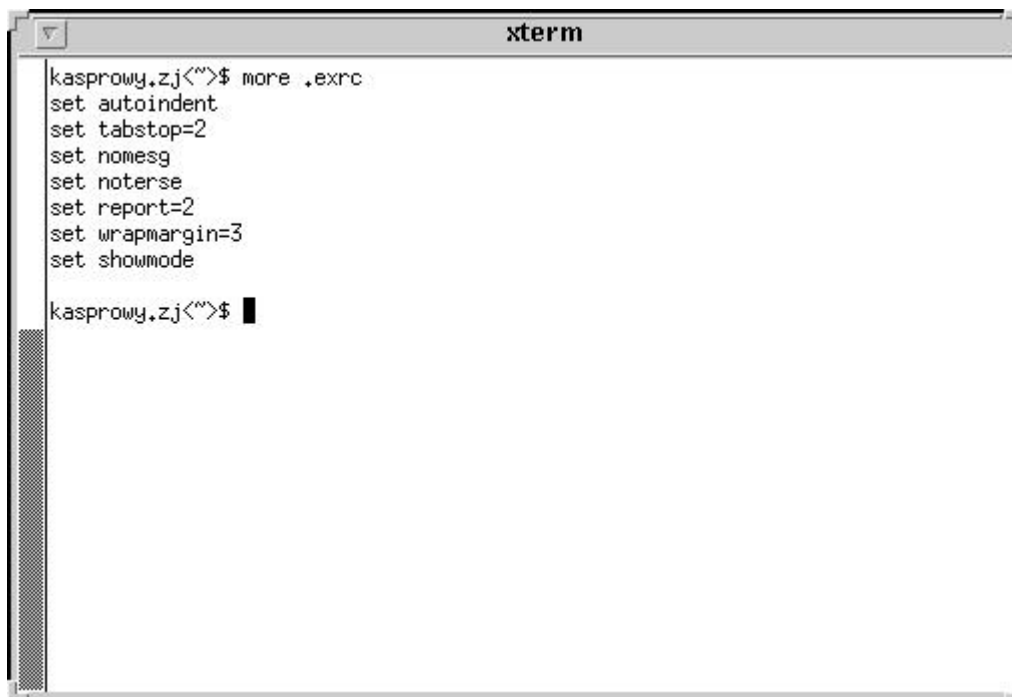


```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4
5 entity Counter2 is
6 generic (delay: time := 1 ns);
7 port ( reset: in bit;
8       load : in bit;
9       clk  : in bit;
10      zero : out bit;
11      data : in std_logic_vector ( 0 to 7 );
12      stan: buffer std_logic_vector ( 0 to 7 ));
13 end Counter2;
14
15 architecture Behav of Counter2 is
16 begin
17   LICZ: process (reset, clk)
18   begin
19     if reset = '0' then
20       stan <= "00000000" after delay;
21     elsif (clk'event and clk = '1' ) then
22       if load = '1' then
23         stan <= stan - 1;
24       else
25         stan <= data;
26       end if;
27     end if;
28   end process LICZ;
29
30   With stan select
31     zero <= '0' when "00000000",
32            '1' when others;
33
34
35
36 end Behav;
```

7 substitutions on 6 lines

Rys. 3.10 Edytor **vi** - stan po wykonaniu polecenia :%s/count/stan/g

Użytkownik może dostroić zachowanie edytora **vi** poprzez włączenie odpowiednich opcji, takich jak np. **nu**. Oczywiście wydawanie tych poleceń przy każdym uruchomieniu edytora byłoby pozbawione sensu. Opcje ustawia się w pliku konfiguracyjnym (*ang. dot file*) o nazwie **.exrc**. Jest to tzw. plik ukryty i musi się znajdować w katalogu domowym użytkownika. Jest to typowy dla systemu Unix sposób konfigurowania programów użytkowych. W momencie uruchamiania edytora plik **.exrc** jest odczytywany a zawarte w nim polecenia automatycznie wykonywane. Na rysunku 3.11 pokazano przykładową zawartość takiego pliku. Zachęcamy do praktycznego sprawdzenia, jakie są efekty włączenia tych opcji. Jak zawsze więcej informacji o edytorze **vi** można uzyskać za pomocą polecenia **man vi**.



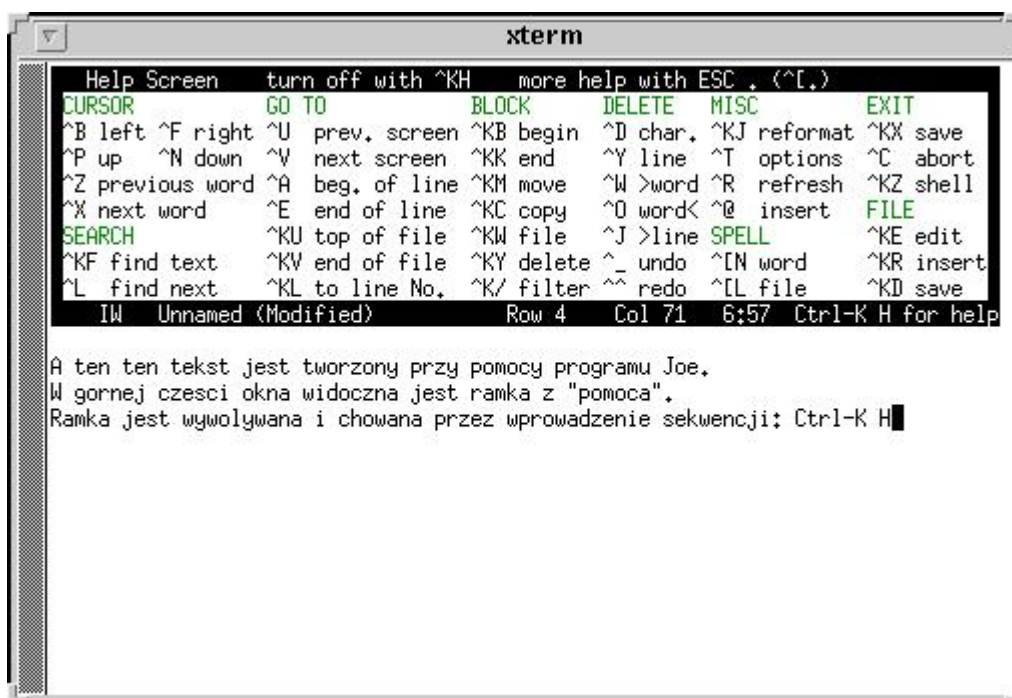
```
kasprowy.zj<^>$ more .exrc
set autoindent
set tabstop=2
set nmesg
set noterse
set report=2
set wrapmargin=3
set showmode
kasprowy.zj<^>$ █
```

Rys. 3.11 Przykładową zawartość pliku **.exrc**

*Trzeba pamiętać, że w niektórych awaryjnych sytuacjach nie pomoże nam nawet edytor **vi**. Wynika to z faktu, że program **vi** jest edytorem pełnoekranowym i do jego prawidłowej pracy wymagana jest poprawna definicja terminala. W takiej sytuacji należy się uciec się do pomocy edytora **ed**, który jest edytorem wierszowym i działa w każdych warunkach (więcej informacji dostarczy polecenie **man ed**).*

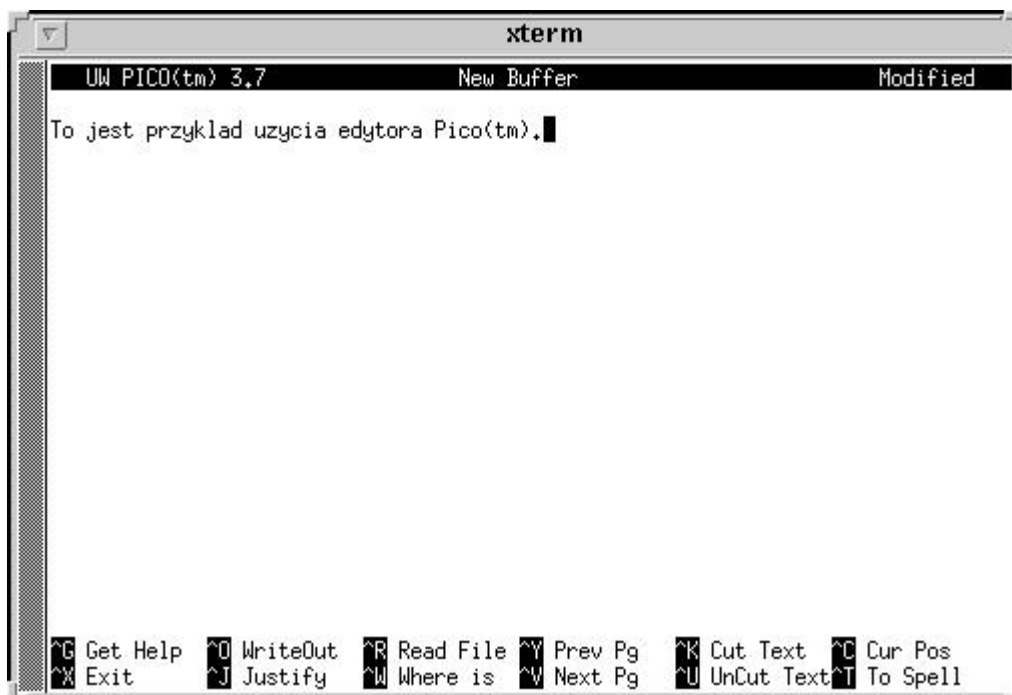
Edytory pracujące w trybie tekstowym - Joe, Pico i wbudowany programu Midnight Commander

Na rysunku 3.12 można znaleźć przykład krótkiej sesji z edytorem **joe**. Edytor ten reprezentuje koncepcję typową dla systemu MS-DOS. Brak jest oddzielnych trybów pracy a polecenia wydaje się za pomocą specjalnych sekwencji zaczynających się od znaku **Ctrl**. Obsługa programu jest w zasadzie oczywista i nie będzie tu omawiana.



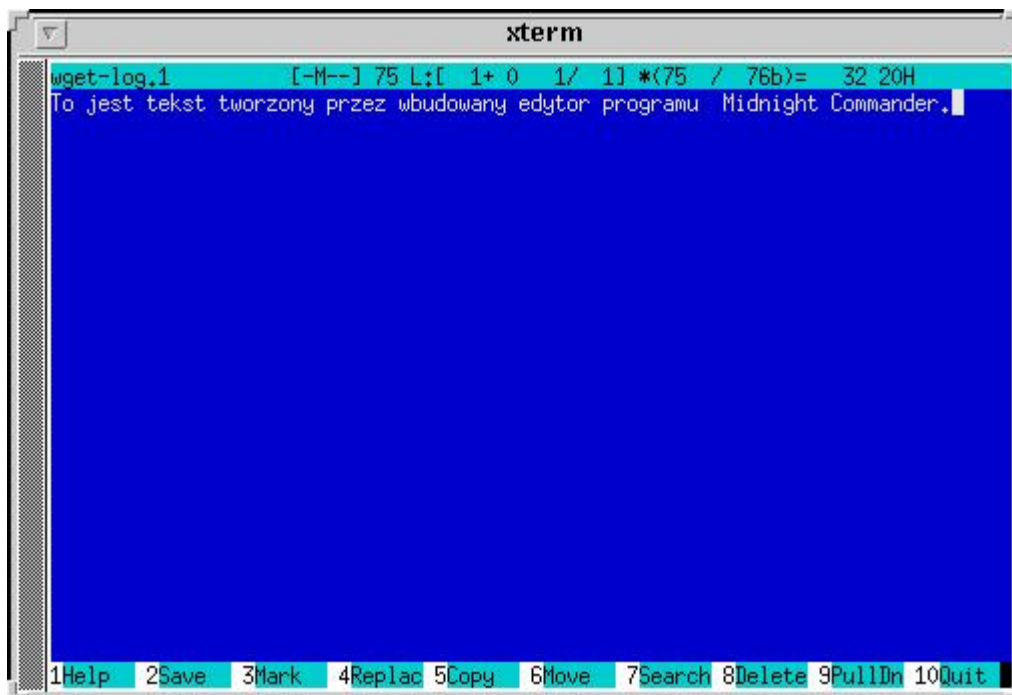
Rys. 3.12 Przykład pracy z edytor **Joe**

Na rysunku 3.13 pokazano edytor **Pico**, który jest dostarczany wraz z programem pocztowym **Pine**. Edytor należy do tej samej kategorii co Joe i podobnie jak w poprzednim przypadku obsługa programu jest oczywista i nie będzie omawiana.



Rys. 3.13 Przykład pracy z edytor **Pico**

Rys. 3.14 pokazuje przykład pracy z edytorem wbudowanym do programu Midnight Commander.



Rys. 3.14 Przykład pracy z edytorem wbudowanym do programu Midnight Commander

Edytory pracujące w trybie graficznym

Przejdziemy teraz do prezentacji edytorów działających w trybie graficznym: **gvim**, **Nedit** i **Xemacs**. Pierwszy z nich - **gvim** - to edytor **vim** wyposażony w graficzny interfejs użytkownika. Odkrywanie jego wad i zalet pozostawiamy zainteresowanym ;-)

Autorzy rekomendują natomiast wszystkim zapoznanie się z dwoma bardzo wygodnymi programami **Nedit** i **Xemacs** (rys. 3.15 i 3.16). Programy te można polecić szczególnie programistom, gdyż posiadają one szereg cech ułatwiających pracę (np. wyróżnianie słów kluczowych). Na rysunkach 3.15 i 3.16 pokazano przykład edycji modelu VHDL prostego licznika, choć oczywiście programy te rozpoznają składnię wielu innych języków programowania. Program **Nedit** (www.nedit.org) jest stosunkowo prostym edytorem, choć wyposażonym w praktycznie wszystkie potrzebne programiście cechy. Natomiast **XEmacs** czyli "okienkowa" wersja edytora **Emacs** to chyba najwszechstronniejszy edytor jaki kiedykolwiek powstał. Należy on do pakietu GNU - narzędzi opracowanych przez Free Software Foundation (www.gnu.org). Jego możliwości wykraczają daleko poza cechy zwykłego edytora. Po odpowiednim skonfigurowaniu może on np. pełnić funkcję programu pocztowego lub kalkulatora.

```
counter2.vhd
File Edit Search Preferences Shell Macro Windows Help
Find:
Rev Literal Case RegExp
/users/zj/Dydaktyka/lab4/counter2.vhd line 1, col 0, 731 bytes

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

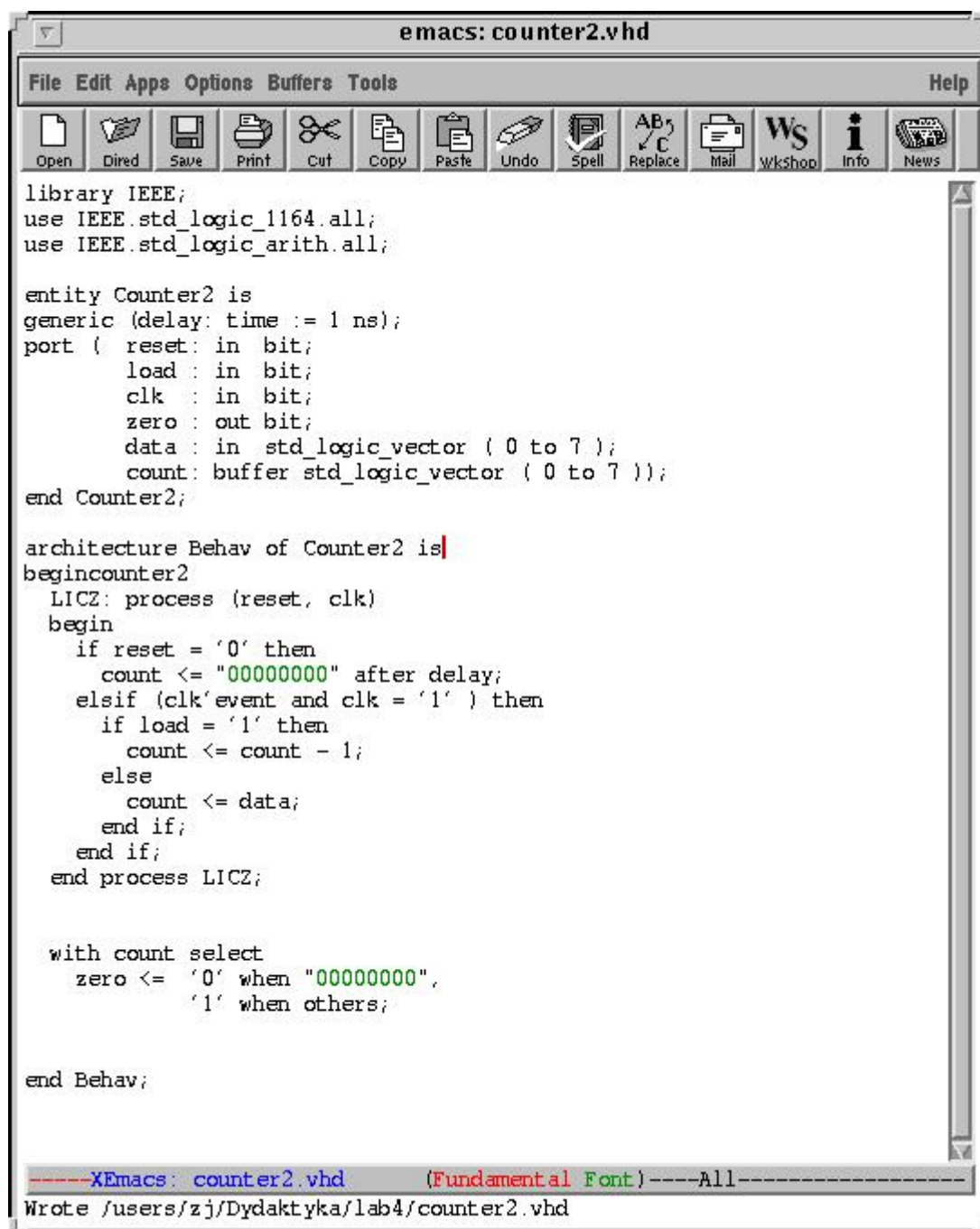
entity Counter2 is
generic (delay: time := 1 ns);
port (
    reset: in bit;
    load : in bit;
    clk  : in bit;
    zero : out bit;
    data : in std_logic_vector ( 0 to 7 );
    count: buffer std_logic_vector ( 0 to 7 ));
end Counter2;

architecture Behav of Counter2 is
begin
    LICZ: process (reset, clk)
    begin
        if reset = '0' then
            count <= "00000000" after delay;
        elsif (clk'event and clk = '1' ) then
            if load = '1' then
                count <= count - 1;
            else
                count <= data;
            end if;
        end if;
    end process LICZ;

    with count select
        zero <= '0' when "00000000",
               '1' when others;

end Behav;
```

Rys. 3.15 Przykład zastosowania edytora Nedit



Rys. 3.16 Przykład zastosowania edytora XEmacs

3.5. Archiwizowanie i kompresja plików

Typowe rozwiązania problemu archiwizacji i kompresji plików w systemach typu Unix różni się dość znacznie od podobnych czynności wykonywanych w systemach z rodziny Windows. Podstawowa różnica polega na tym, że są one wykonywane jako oddzielne operacje realizowane przez różne polecenia. Stosowane też są inne formaty i algorytmy.

W ostatnich latach, wraz z rozpowszechnianiem się systemu Linux, pojawiły się programy usługowe ułatwiające wykonywanie tych czynności. Przeniesiono też na platformę Unix programy typowe dla systemów MS Windows.

W lekcji tej przedstawione zostaną standardowe narzędzia do archiwizacji i kompresji plików dostępne we wszystkich systemach typu Unix oraz oprogramowanie dostępne w systemie Linux.

Kompresja

Do kompresji plików w systemach typu Unix stosowany jest standardowo program **compress**. Dekompresję plików można wykonać przy pomocy programów **uncompress**. Ponadto, dostępny jest program pomocniczy **zcat** umożliwiający wyświetlenie zawartości plików bez konieczności ich uprzedniego dekompresowania.

Składnia poleceń **compress**, **uncompress** i **zcat** jest następująca:

```
compress [-c] [-f] [-v] [plik...]  
uncompress [-c] [-f] [-v] [plik...]  
zcat [plik...]
```

gdzie:

- c** - wynik działania przekazywany na **stdout**; plik oryginalny nie jest usuwany,
- f** - tryb forsowny,
- v** - powoduje wypisanie dodatkowych komunikatów,

plik - nazwa pliku do (de)kompresowania.

Program **compress** dokonuje zmniejszenia objętości pliku stosując algorytm LZW (algorytm ten został opatentowany). W wyniku jego działania tworzony jest nowy plik (o ile nie zastosowano opcji **-c**) o nazwie powstałej poprzez dodanie do nazwy oryginału końcówki **.Z**. Plik oryginalny jest usuwany. Jeśli w linii wywołania nie podano argumentu **plik** to dane są czytane ze standardowego wejścia.

Program **uncompress** odtwarza oryginalną zawartość pliku skompresowanego uprzednio przy pomocy programu **compress**. W wyniku działania tworzony jest nowy plik o nazwie powstałej przez usunięcie z nazwy pliku wejściowego końcówki **.Z**. Plik wejściowy jest usuwany. Jeśli zastosowano opcję **-c** to wynik działania **uncompress** przekazywany jest standardowe wyjście. Jeśli w linii wywołania nie podano argumentu **plik** to dane są czytane ze standardowego wejścia a wynik przekazywany jest na standardowe wyjście.

Polecenie **zcat** działa tak jak **uncompress** z opcją **-c**.

Przykład

```
compress *.c  
uncompress *.c.Z
```

W systemie Linux standardowymi narzędziami do kompresji plików są programy z pakietu GNU: **gzip**, **gunzip**, **zcat**, **zmore**, **zgrep**, **zdiff**, **zcmp**.

Oprogramowanie to nie jest dostarczane w dystrybucjach systemów komercyjnych takich jak np. SUN Solaris. W praktyce jednak większość instalacji tych systemów posiada je jako oprogramowanie opcjonalne.

Składnia poleceń jest następująca (pokazano tylko ważniejsze opcje):

```
gzip [-c] [-d] [-f] [-r] [-v] [plik...]
```

```
gunzip [-c] [-f] [-v] [plik...]
```

```
zcat [-f] [plik...]
```

gdzie:

-c - wynik działania przekazywany na **stdout**; plik oryginalny nie jest usuwany,

-d - włącza tryb dekompresji,

-f - tryb forsowny,

-r - powoduje rekurencyjne przeglądanie drzewa katalogów i (de)kompresowanie znajdujących się tam plików

-v - powoduje wypisanie dodatkowych komunikatów,

plik - nazwa pliku do (de)kompresowania.

Program **gzip** dokonuje zmniejszenia objętości pliku i zastępuje go przez nowy plik (o ile nie zastosowano opcji **-c**) o nazwie powstałej poprzez dodanie do nazwy oryginału końcówki **.gz** (ze względu na zastrzeżenia patentowe dotyczące algorytmu LZW stosowany jest tu inny algorytm o nazwie LZ77). Jeśli w linii wywołania nie podano argumentu **plik** to dane są czytane ze standardowego wejścia.

Program **gunzip** odtwarza oryginalną zawartość pliku skompresowanego uprzednio przy pomocy programu **gzip** lub programu **compress**. W wyniku działania tworzony jest nowy plik o nazwie powstałej przez usunięcie z nazwy pliku wejściowego końcówki **.gz** (lub **.Z**). Plik wejściowy jest usuwany. Jeśli zastosowano opcję **-c** to wynik działania **gunzip** przekazywany jest standardowe wyjście.

Zamiast polecenia **gunzip** można zastosować polecenie **gzip** z opcją **-d**. Jeśli w linii wywołania nie podano argumentu **plik** to dane są czytane ze standardowego wejścia a wynik przekazywany jest na standardowe wyjście.

Polecenie **zcat** (w wersji GNU) działa tak jak **gunzip** z opcją **-c**.

Przykład

```
gzip *.c
gunzip *.c.gz
gzip -c file1 > foo.gz
cat file1 file2 | gzip > foo.gz
gzip -cd file.gz | wc -c
```

Dodatkowo dostępne są programy pomocnicze **zmore**, **zgrep**, **zdiff**, **zcmp** umożliwiające wyświetlenie zawartości, wyszukiwanie wzorców lub też porównywanie plików bez konieczności ich uprzedniego dekompresowania. Mają one następującą składnię:

```
zmore [ plik... ]
```

```
zgrep [ opcje_polecenia_grep ] wzorzec [plik...]
```



```
zcmp [ opcje_polecenia_cmp ] plik1 [ plik2 ]
```

```
zdiff [ opcje_polecenia_diff ] plik1 [ plik2 ]
```

Programy te stanowią interfejs pozwalający stosować standardowe polecenia **more**, **grep**, **cmp** i **diff** dla plików skompresowanych za pomocą programów **gzip** i **compress**.

Archiwizacja - program tar

Typowym narzędziem przeznaczonym do archiwizacji plików dostępnym we wszystkich systemach typu Unix jest program **tar**.

Program **tar** tworzy archiwum w postaci strumienia danych, który to strumień zawiera wszystkie pliki wskazane w wierszu wywołania. Archiwizowana jest nie tylko zawartość plików, ale także ich atrybuty takie jak prawa dostępu, właściciel, grupa, data modyfikacji, itp. Jeśli w wywołaniu podano nazwę katalogu to archiwizacji podlegają wszystkie pliki znajdujące się w poddrzewie systemu plików, którego korzeniem jest wskazany katalog. W tym przypadku nazwy plików zawierają informację o swoim położeniu w drzewie. Podczas ekstrakcji plików z archiwum odtwarzane są oryginalne nazwy plików, zawartość plików oraz ich atrybuty a także tworzone są nowe katalogi o ile jest wymagane. Tworząc archiwum przy pomocy polecenia **tar** należy pamiętać o konsekwencjach sposobu specyfikowania nazw plików przeznaczonych do archiwizacji. Zazwyczaj zaleca się, aby pliki specyfikować za pomocą nazw względnych a nie nazw bezwzględnych (tzn. zaczynających się od znaku */*). Zalecenie to wynika właśnie stąd, że podczas ekstrakcji odtwarzane są oryginalne nazwy plików. Jeśli nazwy te odwołują się do korzenia drzewa katalogów to może się okazać konieczne utworzenie plików/katalogów w katalogach, do których dany użytkownik nie posiada prawa do zapisu (np. */*). Oczywiście w takiej sytuacji ekstrakcja plików będzie niemożliwa. Jeśli jednak zastosujemy nazwy względne to ekstrahowane pliki będą umieszczane względem bieżącego katalogu.

Program **tar** umożliwia wykonywanie archiwów w postaci plików dyskowych lub zapisywanych na innych nośnikach takich jak np. taśmy (*ang. streamer*) czy dyskietki.

Polecenie **tar** nie dokonuje kompresji archiwizowanych plików. O ile archiwum nie jest tworzone na taśmie, kompresja archiwum wykonywana jest w odrębnym kroku przy pomocy programów **compress** czy **gzip** w taki sam sposób jak każdego innego pliku. W przypadku wykonywania archiwum na taśmie należy dokonać kompresji danych przed ich zapisaniem. Przykłady takich operacji podamy w dalszej części tej lekcji.

Składnia wywołania programu **tar** jest następująca (pokazano tylko najważniejsze opcje):

```
tar [-] [c|t|x] [v] [b N] [B] [f archiwum] [plik | katalog]...
```

gdzie:

- c** - tworzy nowe archiwum,
- t** - wypisuje zawartość archiwum,
- x** - ekstrahuje plik(i),
- v** - powoduje wypisanie dodatkowych komunikatów,
- b** - definiuje rozmiar bloku danych jako $N \cdot 512$ bajtów (domyślenie $N=20$),
- B** - wymusza odczytanie całego bloku danych; opcja stosowana przy współpracy z łączami,
- f** - za tą opcją musi znaleźć się nazwa archiwum lub znak - reprezentujący **stdin** lub **stdout**,
- plik | katalog** - pliki do zarchiwizowania/ekstrakcji.

W wywołaniu programu **tar** można pominąć znak "-" stawiany przed grupą znaków reprezentujących opcje.

Opcje należy zestawiać w grupy w zależności od wykonywanej czynności. Przy czym pierwszą opcją **musi** być jedna z **c,t,x**. I tak, w przypadku tworzenia archiwum w postaci pliku dyskowego należy zastosować zestaw opcji **-cf** (dodatkowa opcja **v** powoduje wyłączenie zwiększenia ilości informacji wypisywanej na terminalu), np.:

```
tar -cf backup.tar ./plik1 ./plik2 ./katalogA ./katalogB
```

Jeśli te same pliki mają być zarchiwizowane na taśmie to należy użyć tych samych opcji, ale nazwę archiwum trzeba zastąpić nazwą pliku specjalnego reprezentującego dane urządzenie, np.:

```
tar -cf /dev/rmt0 ./plik1 ./plik2 ./katalogA ./katalogB
```

Jeśli przed zapisaniem na taśmie archiwum ma być poddane kompresji to należy :

1. skierować strumień danych produkowany przez program **tar** na jego standardowe wyjście przez zastąpienie nazwy archiwum znakiem "-" (minus),
2. skompresować strumień za pomocą wybranego programu, np. **gzip**,
3. skierować strumień skompresowanych danych do pliku specjalnego reprezentującego dane urządzenie za pomocą polecenia **dd**.

Przykład takiego polecenia podajemy poniżej:

```
tar -cvf - ./plik1 ./plik2 ./katalogA ./katalogB | gzip -c | dd  
of=/dev/rmt0
```

Należy pamiętać, aby podczas ekstrakcji plików z tak wykonanego archiwum dokonać wstępnej dekompresji strumienia danych odczytywanego z taśmy.

Wykorzystywanie taśmy jako urządzenia do archiwizacji wymaga dokładniejszego zapoznania się z działaniem poleceń **tar**, **dd** i **mt**. Odpowiednie informacje można uzyskać za pomocą polecenia **man**. Należy też wiedzieć, że w systemie występują dwie wersje pliku specjalnego reprezentującego urządzenie taśmowe: **przewijający** i **nieprzewijający** (ang. *rewinding* i *non-rewinding*), np. **/dev/rmt0** i **/dev/nrmt0**. Różnica polega na tym, że przy wykorzystywaniu pliku przewijającego taśma po zakończeniu każdej operacji zapisu lub odczytu zostanie przewinięta na początek, podczas gdy przy zastosowaniu pliku nieprzewijającego głowica urządzenia ustawiona będzie za ostatnim odczytanym lub zapisanym blokiem danych. Dzięki stosowaniu pliku nieprzewijającego, na jednej taśmie można zapisać więcej niż jedno archiwum, a więc wykorzystać ją bardziej efektywnie.

Jeśli użytkownik chce poznać zawartość archiwum (co jest szczególnie zalecane przed wykonaniem ekstrakcji plików) powinien zastosować opcje **-tf**, np.

```
tar -tvf backup.tar
```

Natomiast ekstrakcja plików z archiwum wymaga zastosowania grupy opcji **-xf**, np:

```
tar -xf backup.tar
```

Jeśli użytkownik chce wyekstrahować tylko wybrane pliki to musi podać ich dokładną nazwę, taką pod jaką występują w archiwum (nazwę tę można poznać wywołując program **tar** z opcjami **-tf**), a więc łącznie ze ścieżką jeśli archiwizacji podlegał cały katalog, np:

```
tar -xf backup.tar ./kasia/projekt/main.c
```

Jeśli archiwum zostało skompresowane to można odczytywać jego zawartość lub wykonywać ekstrakcję plików posługując się poleceniem **zcat** oraz zastępując nazwę archiwum przez znak "-" co powoduje, że **tar** pobiera dane ze standardowego wejścia, np:

```
zcat backup.tar.gz | tar -xvf -
```

Podobna czynność dla przypadku archiwum wykonanego na taśmie wymaga wstępnej dekompresji strumienia danych odczytywanego z taśmy, co można wykonać w sposób następujący:

1. odczytać strumień skompresowanych danych z pliku specjalnego reprezentującego dane urządzenie za pomocą polecenia **dd**,
2. skierować odczytany strumień na standardowe wejście polecenia dekompresującego, np. **gunzip**,
3. strumień zdekompresowany skierować na standardowe wejście polecenia **tar**.

Przykład takiego polecenia podajemy poniżej:

```
dd if=/dev/rmt0 | gunzip -c | tar -xvf -
```

Dostępna jest także wersja GNU programu **tar**. Jest ona standardowym programem użytkowym dostarczającym w dystrybucji systemu Linux. Wersja ta posiada szereg dodatkowych możliwości. Najważniejsze z nich to możliwość automatycznego filtrowania przetwarzanego strumienia danych przez programy do kompresji i dekompresji plików. Opcje włączające te dodatkowe funkcje to:

- Z** - włącza filtrowanie za pomocą **compress** lub **uncompress**,
- z** - włącza filtrowanie za pomocą **gzip** lub **gunzip**,
- I** - włącza filtrowanie za pomocą **bzip2**,

Programy zip i unzip

W systemie Linux oprócz standardowego dla systemów typu Unix programu **tar** dostępne są także programy **zip** i **unzip**. Program **zip** wykonuje jednocześnie archiwizację i kompresję plików, a więc zastępuje dwa standardowe programy systemu Unix **tar** i **compress** (lub **tar** i **gzip**) i jest kompatybilny z programem **PKZIP** opracowanym dla systemu MS-DOS. Program **unzip** jest narzędziem pozwalającym na wyświetlenie zawartości, testowanie oraz ekstrakcję plików z archiwum wykonanego przy pomocy **zip**-a. Dostępne są oczywiście wersje **zip**-a na wiele innych systemów, np.: VMS, OS/2, Unix, Windows NT, Minix, Atari, Mac OS, Amiga i Acorn RISC OS, choć nie konieczne muszą stanowić ich standardowe wyposażenie (np. w SUN Solaris standardowo dostępny jest tylko program **unzip**).

Program **zip** dokonuje kompresji wskazanych plików i tworzy z nich archiwum, zapisując jednocześnie informacje o atrybutach plików takich jak nazwa, położenie w katalogu, data i czas ostatniej modyfikacji, prawa własności i dostępu a także dodatkowe informacje kontrolne pozwalające na stwierdzenie czy archiwum nie jest uszkodzone. Nazwa archiwum tworzona jest z nazwy podanej w linii wywołania i rozszerzenia .zip, chyba że podana nazwa zawiera już kropkę. Składnia polecenia zip jest następująca:

```
zip [opcje] [archiwum [plik...]]
```

Najważniejsze opcje to:

- - zastępuje listę plików do archiwizacji,
- @** - pobiera listę plików do archiwizacji z **stdin**,
- P password** - powoduje użycie hasła **password** do zakodowania archiwum,

- R - archiwizuje całe drzewo podkatalogów zaczynając od bieżącego katalogu,
- T - testuje poprawność struktury archiwum,
- d - usuwa wskazane pliki z archiwum,
- e - koduje zawartość archiwum wykorzystując hasło wprowadzone przez użytkownika w trybie interaktywnym,
- f - powoduje, że plik znajdujący się już w archiwum jest zastępowany tylko wtedy, gdy dodawany plik ma późniejszą datę modyfikacji; nie dodaje plików nie istniejących w archiwum,
- g - powoduje dodawanie do istniejącego archiwum zamiast tworzenie nowego,
- j - zmienia domyślne działania programu zip powodując, że archiwizowane pliki tracą początkową część nazwy wskazującej na położenie w drzewie katalogów a nazwy katalogów nie są zapamiętywane w archiwum,
- k - powoduje, że zip próbuje przetworzyć nazwy plików na zgodne ze standardem MS-DOS,
- l - dokonuje translacji znaku końca wiersza (EOL) ze standardu obowiązującego w systemie Unix (znak LF) na standard obowiązujący w systemie MS-DOS (para znaków CR LF)- **opcja tylko dla plików tekstowych**,
- ll - dokonuje translacji znaku końca wiersza (EOL) ze standardu obowiązującego w systemie MS-DOS (para znaków CR LF) na standard obowiązujący w systemie Unix (znak LF) - **opcja tylko dla plików tekstowych**,
- m - "przenosi" pliki do archiwum, czyli usuwa oryginały po ich dodaniu do archiwum,
- n **rozszerzenie** - nie kompresuje plików zakończonych przez **rozszerzenie**,
- r - archiwizuje całe drzewo podkatalogów zaczynając od wskazanego katalogu,
- u - powoduje, że plik znajdujący się już w archiwum jest zastępowany tylko wtedy, gdy dodawany plik ma późniejszą datę modyfikacji; dodaje pliki nie istniejące dotychczas w archiwum,
- v - powoduje wypisanie dodatkowych informacji,
- x **pliki** - powoduje, że wskazane pliki będą pominięte w trakcie archiwizacji.

Utworzenie archiwum ze wszystkich plików (włącznie z ukrytymi) znajdujących się w bieżącym katalogu wymaga użycia polecenia o postaci:

```
zip archiwum .* *
```

Powstanie wtedy archiwum o nazwie **archiwum.zip**.

W wyniku następnego polecenia powstanie archiwum o nazwie **archiwum.zip** zawierające wszystkie pliki znajdujące się w podrzewie katalogów zaczynającym się od katalogu okno:

```
zip -r archiwum ./okno
```

Natomiast wykonanie polecenia :

```
zip -R foo '*.c'
```

spowoduje stworzenia archiwum o nazwie **foo.zip** zawierającego wszystkie pliki o nazwie pasującej do wzorca ***.c** i znajdujących w podrzewie katalogów zaczynającym się od bieżącego katalogu.

Kolejne polecenie stworzy archiwum o nazwie **archiwum.zip** zawierające wszystkie pliki znajdujące się w podrzewie katalogów zaczynającym się od katalogu projekt z wyjątkiem plików o nazwach pasujących do wzorca ***.o**:

```
zip -r archiwum projekt -x \*.o
```

Następny przykład demonstruje zastosowanie opcji **-@**:

```
find . -name "*.ch" -print | zip archiwum -@
```

Usuwanie plików z archiwum pokazuje kolejny przykład. W wyniku zastosowania tego polecenia z archiwum zostaną usunięte wszystkie pliki, których nazwy zaczynają się od **a/b/** oraz wszystkie te kończące się na ***.o**:

```
zip -d archiwum a/b/\* \*.o
```

Program **zip** może też współpracować z poleceniami **tar** i **dd**, pełniąc podobną rolę jak programy **compress** czy **gzip**. Poniżej prezentujemy kilka przykładów takiej współpracy:

- zapis archiwum bezpośrednio na taśmę:

```
zip -r - . | dd of=/dev/nrst0 obs=16k
```

-kompresja archiwum wykonanego w formacie **tar**:

```
tar cf - . | zip backup -
```

- wykorzystanie programu **zip** jako "filtra kompresującego":

```
tar cf - . | zip | dd of=/dev/nrst0 obs=16k
```

Program **unzip** domyślnie, tzn. przy braku opcji, dokonuje ekstrakcji całej zawartości archiwum w bieżącym katalogu i ewentualnie znajdujących się tam podkatalogach. Wywołanie programu jest następujące:

```
unzip [-opcje] archiwum[.zip] [file...] [-x plik...] [-d katalog]
```

Do najważniejszych opcji należą:

- C** - powoduje, że wielkie i małe litery nie są rozróżniane w nazwach plików,
- P password** - używa hasło **password** do odkodowania archiwum,
- X** - powoduje, że odtwarzane są prawa własności do plików,
- a** - włącza konwersję plików tekstowych,

- b** - powoduje, że wszystkie pliki traktowane są jak binarne,
- c** - powoduje, że ekstrahowane pliki są wysyłane na **stdout**,
- d katalog** - specyfikuje katalog, w którym zapisane mają być wyekstrahowane pliki,
- f** - powoduje, że ekstrahowane są tylko te pliki które już istnieją na dysku (zgłaszane jest żądanie potwierdzenia skasowania istniejących plików),
- j** - powoduje, że w czasie ekstrakcji nie jest odtwarzana struktura drzewa katalogów,
- l** - wypisuje zawartość archiwum,
- n** - powoduje, że w czasie ekstrakcji istniejące pliki nie są kasowane,
- o** - powoduje, że w czasie ekstrakcji wszystkie istniejące pliki są kasowane bez ostrzeżenia,
- p** - powoduje ekstrakcję plików do **stdout**, w przeciwieństwie do opcji **-c** pliki są wysyłane w postaci w jakiej istnieją w archiwum (bez konwersji),
- t** - testuje pliki archiwum,
- u** - działa tak jak opcja **-f** (z żądaniem potwierdzenia skasowania istniejących plików), z tą różnicą że ekstrahowane są też pliki nie znajdujące się na dysku,
- v** - powoduje wypisanie dodatkowych informacji,
- x plik...** - specyfikuje listę plików do pominięcia,
- z** - powoduje wypisanie wyłącznie komentarza archiwum,

Poniższy przykład pokazuje najprostsze wywołania programu **zip** w celu ekstrakcji wszystkich plików z archiwum projekt.zip:

```
unzip projekt.zip
```

Użycie opcji **-j** spowoduje, że wszystkie pliki znajdą się w bieżącym katalogu niezależnie od tego czy w strukturze archiwum istnieją podkatalogi:

```
unzip -j projekt.zip
```

Ekstrakcję pliku w celu przekierowania strumienia danych do łącza można wykonać z pomocą opcji **-p**, tak jak to pokazuje poniższy przykład:

```
unzip -p articles paper1.dvi | dvips
```

Kolejny przykład to ekstrakcja wszystkich plików źródłowych w językach C i FORTRAN, bez różnicowania wielkich i małych liter do innego katalogu niż bieżący:

```
unzip -C source.zip "*.fch" -d /tmp
```

Program **unzip** może też współpracować z poleceniami **tar** i **dd**, pełniąc podobną rolę jak programy **uncompress** czy **gunzip**. Przykłady takiej współpracy pokazują dwa następujące polecenia:

```
unzip -p backup | tar xf -
```

```
dd if=/dev/nrst0 ibs=16k | unzip | tar xvf -
```

Użytkownik może zdefiniować domyślny zestaw opcji dla programów **zip** i **unzip** poprzez utworzenie zmiennych środowiska **ZIPOPT** i **UNZIP** i wpisanie do nich zestawu żądanych opcji.

Bibliografia

1. Glass G., Ables K.: Linux dla programistów i użytkowników, Wydawnictwo Helion 2007 (rozdziały: 3, 4)

Nowe pojęcia, definicje i wyrażenia

Termin	Objaśnienie
dowiązanie	zapis w katalogu wskazujący na fizyczne położenie pliku
dowiązanie symboliczne	plik wskazujący na położenie innego pliku w strukturze katalogowej
plik	logiczna jednostka przechowywania informacji w pamięci pomocniczej
prawa dostępu	atrybut pliku określający uprawnienia w dostępie do pliku dla właściciela, wyróżnionej grupy użytkowników i pozostałych
prawa własności	atrybuty pliku określające identyfikatory właściciela pliku i wyróżnionej grupy użytkowników
system plików	mechanizm bezpośredniego przechowywania i dostępu do informacji w systemie operacyjnym
typ pliku	atrybut pliku określający jego przeznaczenie i sposób interpretacji zawartości przez system operacyjny