

1 Przypomnienie

Podstawowe katalogi i ich funkcja

/bin - tu zapisywane są wszystkie podstawowe pliki bez których niemożliwy jest start i późniejsza praca systemu

/boot - zawiera jądro systemu oraz pliki niezbędne do poprawnego uruchomienia i pracy systemu

/dev - zgodnie z zaimplementowaną w systemach unixowych koncepcją urządzeń systemowych katalog ten zawiera pliki reprezentujące poszczególne komponenty sprzętowe naszej maszyny

/etc - tu są zapisywane pliki konfiguracyjne wszystkich zainstalowanych w systemie programów

/home - tu tworzone są katalogi domowe użytkowników. Katalog ten powinien znajdować się na innej partycji.

/lib - gromadzi biblioteki współdzielone przez zainstalowane w systemie aplikacje

/lost+found - tu zapisywane są pliki i katalogi tworzone podczas naprawiania systemu plików

/mnt - autorzy systemu proponują właśnie tu podmontowywać zewnętrzne systemy plików

/opt - odpowiednik Program Files w MS Windows

/proc - podczas pracy system tworzy tu pliki umożliwiające kontrolę jego działania

/root - katalog osobisty superużytkownika

/tmp

/usr - bardzo duży objętościowo katalog zawierający pliki i biblioteki tworzące środowisko X-Window i inne podstawowe aplikacje unixowe, t.j. grep, awk, more, less itd. (podkatalog /bin)

/var - tutaj przechowywane są pliki dzienników systemowych (/log), w których zapisywane są poszczególne zdarzenia. Tu także trafia poczta przychodząca do użytkowników systemu oraz tworzone są kolejki poczty wychodzącej i drukowania

Przykład wykorzystania potoku

```
ls --help | more
```

Przekierowanie standardowego we-wy

```
echo "Ala ma kota" > alafile
echo "i psa" >> alafile
```

2 Programowanie w Shellu - Podstawy

Pisanie skryptu polega na łączeniu poleceń shella tak, aby zrealizowane były potrzeby użytkownika. Użytkownik tworzy skrypt za pomocą wpisania poleceń shella do pliku. Można przy tym używać dowolnego edytora, np. vi, emacs lub edytora z programu Midnight Commander. Plik musi być wykonywalny, zatem należy mu ustawić odpowiednie prawa dostępu np.

```
chmod 755 skrypt1
```

2.1 Kilka podstawowych poleceń wykorzystywanych przy pisaniu skryptów

Uwaga!!! Wszystkie poniższe polecenia należy przetestować pisząc odpowiednie skrypty

Polecenie grep szuka określonych napisów w plikach i wyświetla je na standardowym wyjściu.

```
grep "i psa" alafile grep -n "i psa" alafile
```

Zliczanie słów i wierszy, przykład sprawdzający ilu użytkowników pracuje w systemie

```
who | wc -l
```

Znajdź nazwisko użytkownika np. g3 w pliku /etc/passwd za pomocą polecenia

```
grep g3 /etc/passwd
```

Polecenia cut i paste

Wycinanie potrzebnych pól (z wyniku wyświetlonego przy użyciu poprzedniego polecenia) za pomocą polecenia cut (**pola są ograniczone za pomocą znaku :**). Opis pól: 1 - nazwa konta, 2 - hasło itd.

Interesuje nas pozycja 1 i 5, więc

```
grep student /etc/passwdgrep | cut -f1,5 -d:
```

Zakres pól wskazuje się za pomocą znaku -.

Z wyjścia polecenia ls wytnij prawa dostępu i nazwę pliku

```
ls -l | cut -c1-9,55-
```

Zapoznać się z poleceniem `paste` (`man paste`). Napisać skrypt wykorzystujący to polecenie.

Przetwarzanie informacji

Zamiana napisu `ala` na `heniek` w pliku `alafile`

```
sed -e "s/ala/heniek/" alafile
```

Zamiana znak po znaku

Zamiana małych liter na duże z pliku `ala` do pliku `ALA`

```
tr "[a-z]" "[A-Z]" < ala > ALA
```

zamiana spacji na entery

```
tr "[ ]" "[\012]" < ala
```

usuwanie (squeeze) nadmiarowych spacji itp.

```
tr -s "[ ]" "[ ]" < ala
```

usunięcie tabulatora

```
tr -d "[\t]" < ala
```

używanie `awk`, kilka prostych poleceń

```
awk -F: '{print $1 $5}' /etc/passwd  
ls -l | awk -F: '{print $1}'
```

2.2 Trochę o zmiennych

Zmienna reprezentująca ścieżkę do katalogu osobistego użytkownika

`$HOME` lub `~`

Odnajdywanie poleceń przez shell zmienna `$PATH`.

Zapoznaj się z zawartością pliku `.bash_profile`. Zmodyfikuj zmienną `$PATH` tak, aby był widoczny katalog `bin` znajdujący się w twoim katalogu osobistym umieść nim któryś z wcześniej stworzonych plików, a następnie spróbuj go uruchomić z innego katalogu.

Zmienne specjalne - dostarczają informacji o procesie shella. Są one zawsze ustawiane przez shell - ich zawartość jest dostępna po odwołaniu się do nich za pomocą znaku `$`. Wartości tych zmiennych nie można zmieniać za pomocą zwykłych poleceń. np. zmiennym `$0` do `$9` przekazywane są parametry przekazywane do skryptu.

`$#` - nazwa programu shellowego

\$1 . . . - argumenty pozycyjne

\$* - rozwijane do "\$1 \$2 . . ."

\$@ - rozwijane do "\$1" "\$2" . . .

\$\$ - numer procesu bieżącego procesu shella

\$! - numer procesu ostatniego zadania shellowego

Zmienne środowiska - wykorzystywane przez shell do przechowywania danych pomocnych do nadzorowania sesji shella. Użytkownik ma dostęp do nich i może je zmieniać.

\$PS1 - znak zachęty

\t - bieżący czas

\d - bieżąca data

\w - bieżący katalog roboczy

\u - nazwa użytkownika

\h - nazwa hosta

HISTFILE - nawa pliku z historia poleceń

HISTSIZE - liczba pamiętanych poleceń

Wyświetl zawartość zmiennej \$PS1, tzn. `echo $PS1`.

Zmienne programowe

Przykład

```
temp_name=/usr/tmp
cd $temp_name
```

2.3 Instrukcje warunkowe

Polecenie test

Podstawowe warunki testujące (\$? - kod powrotu ostatniego polecenia)

-r plik - prawda, jeśli plik istnieje i ma prawa do czytania

-w plik - prawda, jeśli plik istnieje i ma prawa do pisania

-x plik - prawda, jeśli plik istnieje i ma prawa do pisania

-f plik - prawda, jeśli plik istnieje i jest zwykłym plikiem
-d plik - prawda, jeśli plik istnieje i jest katalogiem
-s plik - prawda, jeśli plik istnieje i ma rozmiar większy od zera
-z s1 - prawda, jeśli długość napisu s1 wynosi zero
-n s1 - prawda, jeśli długość napisu s1 jest różna od zera
s1 = s2 - prawda, jeśli s1 i s2 są identyczne
s1 != s2 - prawda, jeśli s1 i s2 są różne
l1 -eq l2 - równe liczby
l1 -ne l2 - różne
l1 -gt l2 - l1>l2
l1 -ge l2 - l1>=l2
l1 -lt l2 - l1<l2
l1 -le l2 - l1<=l2

Powyższe wyrażenia można łączyć za pomocą operatorów

AND (-a)
OR (-o)

Możliwe jest także użycie operatora NOT (!). Przykłady:

```
test -r filename
echo $?
0
```

```
dwa napisy
test "myname" = "lja"
echo $?
1
```

```
test -w ala -a -r ala
echo $?
```

Współpraca polecenia test z instrukcją warunkową if

```
if [ -r ala ]
then
    echo "ala ma kota"
fi
```

Sprawdzanie czy danemu parametrowi przypisano wartość

```
if [ "$PATH" ] then
    echo $PATH
else
    echo "brak"
fi
```

Sprawdzanie czy parameter jest katalogiem lub plikiem

```
if [ -d ${zmienna} ] then
    przetwarzaj-katalog ${zmienna}
elif [ -f ${zmienna} ] then
    przetwarzaj-plik ${zmienna}
else
    echo "blad"
fi
```

Instrukcja case

```
ala=1
case $ala in
1)
    echo "jeden"
    ;;
2)
    echo "dwa"
    ;;
*)
    echo "nic"
    ;;
esac
```

2.4 Pętle

Przetwarzanie wszystkich plików

```
for $file in * do
    echo $file
done
```

Operacje arytmetyczne na zmiennych

```
month=10
while [ $month -gt 4 ] do
    month=`expr $month -1`
done
```

Napisz przykład wykorzystujący pętlę `until`

```
until [ warunek zakonczenia ] do
    ...
done
%
```

2.5 Dodatkowe polecenia

Wykonywanie poleceń w tle

```
ls -l > heniek & ls -Rl / > heniek & ls -Rl / 1> heniek 2> null &
```

Zapoznać się z poleceniem `noghup`. Co wykonuje poniższa instrukcja?

```
noghup ls -Rl 1> heniek 2> null &
```

Następnie wykonaj instrukcję `logout`, zaloguj się ponownie i sprawdź czy proces odpowiadający poleceniu `noghup` dalej funkcjonuje (polecenie `ps -A`). Jeśli tak to go przerwij (`kill numer_procesu`)

Uruchamianie poleceń o określonym czasie

```
at 10am ls -l
```

3 Zadania

1. Napisz polecenie które będzie sprawdzać czy dana nazwa jest nazwa katalogu.
2. Napisz polecenie `case` sprawdzające czy wartością zmiennej jest `data`, `source`, `comments` lub coś innego.

3. Napisz instrukcję `for` służącą do wyświetlania nazw katalogów dużymi literami.
4. Użyj polecenia `xargs (man xargs)` do przetworzenia wszystkich plików w katalogu. Operację przetwarzania zaimplementuj dowolnie.
5. Za pomocą polecenia `find` odszukaj wszystkie pliki, których właścicielem jest dany użytkownik i które mają nazwę `*.c`. Nazwa użytkownika ma być podawana jako parametr skryptu.
6. Napisz skrypt działający podobnie jak instrukcja `ls -l`. Różnica ma polegać na tym, że prawa dostępu mają być wyświetlane w postaci liczb w formacie ósemkowym.
7. Jaki znak służy do przesyłania poleceń do tła?
8. Jakie polecenie pozwala użytkownikowi shella pozostawić polecenie wykonywane w tle i zakończyć sesję?
9. Napisz polecenie, które będzie przeszukiwało katalogi użytkownika i usuwało pliki ze smieciami (rozszerzenie `.tmp`). Upewnij się, że polecenie będzie wykonane po godzinie 10 wieczór, aby nie obciążać systemu.
10. Napisz polecenie, które będzie edytowało wszystkie pliki w bieżącym katalogu i zastępowało w nich wybrane słowo innym słowem.
11. Napisz skrypt, który będzie sprawdzał czy w wierszu wywołania wpisano argumenty `arg1`, `arg2`, `arg3`. Jeżeli nie, skrypt ma prosić o wprowadzenie tych danych.
12. Napisz skrypt, który będzie sprawdzał opcje postaci: `-c`, `-d`, `-e`. Przypisz argumentom o takiej samej nazwie (`c,d,e`) wartości PRAWDA (1) lub FAŁSZ (0) w zależności od tego czy podano argument w wierszu wywołania.
13. Napisz skrypt, który będzie przeglądał w pętli argumenty wiersza poleceń i przetwarzał je tylko wtedy, gdy są nazwami poleceń.
14. Połącz skrypty z ćwiczeń 6 i 7 w jeden skrypt, który będzie przeglądał w pętli argumenty poprzedzone minusem, np. `-c`, przesuwał je i następnie przeglądał w pętli pozostałe argumenty, przetwarzając je tylko wtedy, gdy będą nazwami plików. Przetwarzanie zaimplementuj w dowolny sposób.
15. Napisz pętlę nieskończoną, w której użytkownik terminala proszony jest o nazwy plików do usunięcia. Pliki te mają być usuwane. Użyj polecenia `trap (man trap)` do eleganckiego zakończenia zadania po jego wykonaniu.
16. Napisz polecenie `trap`, które będzie

- ignorować sygnały,
- usuwać pliki tymczasowe w wypadku otrzymania sygnału SIGQUIT lub SIGINT,
- usuwać pliki tymczasowe w przypadku normalnego zakończenia skryptu.