Przypomnienie komend linux'a.

Krzysztof Dynowski

Komendy podstawowe
date – pokazuje datę i czas systemowy
history – pokazuje historię komend z linii poleceń
pwd (print working directory) – pokazuje pełną ścieżkę do katalogu roboczego(bieżącego)
cd (change directory) – zmiana katalogu roboczego
cd;cd $HOME;cd ~; przejdź do kat. HOME
cd /; przejdź do kat. głównego
mkdir (make directory) – tworzy katalog
mkdir kat1 kat2; utwórz katalogi kat1 i kat2
rmdir (remove directory) – usuwa katalog (katalog musi być pusty), wygodniej *rm -r*
rmdir kat1 kat2; usuń katalogi kat1 i kat2
ls (list) – pokazuje zawartość katalogu roboczego lub podanego w argumencie
ls; listuj katalog roboczy (pliki zaczynające się od '.' są ukryte)
ls -l; listuj katalog roboczy (pliki wyświetlane w formie długiej)
ls -a; listuj katalog roboczy (wyświetlane są wszystkie pliki)
ls -al; listuj katalog roboczy (wszystkie pliki, forma długa)
cp (copy) – kopiuje pliki i/lub katalogi
cp plik1 plik2; kopiuj plik1 na plik2
cp plik kat; kopiuj plik1 do katalogu kat
cp kat1 kat2; kopiuj katalog kat1 do katalogu kat2
cp -a kat1 kat2; kopiuj katalog kat1 do katalogu kat2 (z zachowaniem praw)
mv (move) – przenosie pliki pomiędzy katalogami (też do zmiany nazwy pliku)
mv plik1 plik2; przenieś plik1 na plik2 (zmiana nazwy pliku)
mv plik kat; przenieś plik1 do katalogu kat
mv kat1 kat2; przenieś katalog kat1 do katalogu kat2
rm (remove) – usuwa pliki i/lub katalogi
rm plik1 plik2; usuń plik1 i plik2
rm -r kat; usuń katalog kat wraz z podkatalogami
rm -rf kat; usuń katalog kat wraz z podkatalogami (w trybie force)
rm -ri kat; usuń katalog kat wraz z podkatalogami (w trybie interaktywnym)
chmod (change mode) – ustawia prawa dostępu do pliku/katalogu
chmod +rwx plik; ustaw prawa odczytu,zapisu,wykonywania dla wszystkich(user,group,other)
chmod go-rwx plik; usuń prawa odczytu,zapisu,wykonywania dla grupy i innych
more,less (paged file viewer) – pokazuje zawartość pliku z podziałem na strony
less plik; wyświetl plik (less obsługuje PgUp,PgDn)
more plik; wyświetl plik (tylko w przód)
head,tail (show head,tail of file) – pokazuje początek/koniec pliku
head -10 plik; pokaż 10 początkowych linii z pliku
tail -10 plik; pokaż 10 końcowych linii z pliku
tail -f plik; pokaż końcowe linie z pliku i czekaj na nowe linie (wyjście Ctrl-C)
cat (concatenate files) – skleja podane pliki
cat plik1 plik2; - sklej pliki, wynik na ekran
cat plik1 plik2>plik3; - sklej pliki, wynik do plik3
wc (word count) – zlicza znaki, słowa i linie
wc plik1 plik2; - zlicz znaki, słowa i linie dla podanych plików
file (classify files) – klasyfikuje podane pliki
file /bin/sh; sklasyfikuj podany plik

find (find files) – wyszukuje pliki na dysku wg podanych kryteriów
find /var/www -name ".html";* w katalogu /var/www wyszukaj pliki *.html
locate (locate files) – lokalizuje pliki na dysku (b.szybkie, na podstawie indeksu locatedb)
locate bash; zlokalizuj plik o nazwie bash
grep (regular expression) – przeszukuje pliki wg podanego wyrażenia regularnego
*grep "^A" *.txt;* wyszukaj w plikach *.txt linii zaczynających się od A
sort (sort lines) – sortuje wczytane linie (leksykalnie, numerycznie)
cat plik | sort; sortuj leksykalnie linie z pliku
help (bash help) – pokazuje tekst pomocy dla poleceń powłoki
help for; pokaż informacje o poleceniu *for*
man (manual) – pokazuje tekst pomocy dla programów zewnętrznych
man find; pokaż informacje o programie *find*
man -k editor; wyszukaj manuale zawierające podaną frazę
vi,vim (visual editor) – edytor tekstowy
ESC – przejdź do trybu komend (wyjdź z trybu edycji)
podstawowe komendy edytora:
a,i,A – przejdź do trybu edycji (append,insert,append on line end)
:q – opuść edytor
:q! – opuść edytor (bez zapisania zmian)
:w – zapisz zmiany
:wq – zapisz zmiany i opuść edytor (Shift-ZZ)
diff (get diffirence) – znajduje różnice w plikach tekstowych
diff plik1 plik2; pokaż różnice dla podanych plików
du (disk usage) – pokazuje zajętość podanych katalogów i ich podkatalogów

Komendy archiwizacji plików
df (disk filesystems) – pokazuje zajętość systemu plików (partycji)
mount – pokazuje punktu zamontowania systemów plików (partycji)
gzip (gnu zip) – program pakujący plik
gzip plik1; spakuj plik1 (*plik1* zostanie zastąpiony przez *plik1.gz*)
gunzip (gnu unzip) – program rozpakowujący plik
gzip plik1.gz; rozpakuj plik1.gz (*plik1.gz* zostanie zastąpiony przez *plik1*)
tar (tape archiver) – archiwizator
tar cf arch.tar plik1 plik2 kat1 kat2; archiwizuje podane pliki i katalogi do *arch.tar*
tar xf arch.tar; odzyskuje pliki i katalogi z *arch.tar*
tar tf arch.tar; testuje zawartość archiwum *arch.tar*
tar czf arch.tgz plik1 plik2 kat1 kat2; archiwizuje podane pliki i katalogi do *arch.tgz*
na koniec archiwum zostaje spakowane programem gzip
tar xzf arch.tgz; odzyskuje pliki i katalogi z *arch.tgz*

Komendy pracy w sieci
hostname – pokazuje nazwę hosta (z pliku /etc/hostname)
who,w – pokazuje listę aktualnie zalogowanych użytkowników w systemie
whoami – pokazuje nazwę pod jaką jestem zalogowany do systemu
ping – program badający czas odpowiedzi hosta na pakiety ping (icmp)
ping www.if.pw.edu.pl; zbadaj po nazwie
ping 192.168.1.1; zbadaj po adresie IP
traceroute – program badający ścieżkę do hosta (bazuje na pakietach icmp)
traceroute www.if.pw.edu.pl; zbadaj po nazwie
traceroute 192.168.1.1; zbadaj po adresie IP
host – wyszukuje nazwy w DNS
host www.if.pw.edu.pl; wyszukaj po nazwie

host 194.29.174.1; wyszukaj po adresie IP

ssh – zalogowanie się do systemu zdalnego (bezpieczne połączenie SSL)

ssh *user@student.if.pw.edu.pl;* zaloguj się jako user na serwerze student

Komendy kontroli procesów

ps – pokazuje uruchomione procesy

ps; pokaż moje procesy, widok podstawowy

ps -u user; pokaż procesy użytkownika, widok podstawowy

ps -aux; pokaż wszystkie procesy, widok rozszerzony

top – pokazuje zużycie procesora przez procesy (interaktywny)

kill – wysyła sygnał do procesu o podanym PID (process ID)

kill 1234; wysyła sygnał INT(2) do procesu 1234

kill -KILL 1234; wysyła sygnał KILL(9) do procesu 1234

kill -10 1234; wysyła sygnał USR1(10) do procesu 1234

kill -l; pokazuje listę obsługiwanych sygnałów

su – zalogowanie się jako inny użytkownik

su user; zaloguj na user bez uruchamiania skryptów początkowych

su - user; zaloguj na user z uruchamiania skryptów początkowych

prg& – uruchomienie programu prg w tle

Strumienie wejścia/wyjścia

Standardowy strumień wejścia – klawiatura (0,stdin)

Standardowy strumień wyjścia – ekran (1,stdout)

Standardowy strumień wyjścia błędów – ekran (2,stderr)

Przekierowanie strumienia wyjściowego:

prg > plik.txt; stdout przekierowane do pliku

prg >> plik.txt; stdout dopisane do pliku

prg 2> plik.txt; stderr przekierowane do pliku

prg 2>> plik.txt; stderr dopisane do pliku

prg &> plik.txt; stdout i stderr przekierowane do pliku

Przekierowanie strumienia wejściowego:

prg < plik.txt; stdin przekierowane z pliku

Przekierowanie strumienia wejściowego i wyjściowego:

prg < plikwe.txt>plikwy.txt; stdout dopisane do pliku, stdin przekierowane z pliku

Przetwarzanie potokowe (wyjście jednego programu przekierowane na wejście drugiego):

prg1 | prg2;

Zadania (wykonywać po kolei): czas ok 20min

1. upewnij się że jesteś w kat. domwym (pwd, cd ~)
2. upewnij się że jesteś na właściwym koncie (whoami)
3. wyświetlić zajętość mojego katagu (du -s ., du -s *)
4. utwórz nowy katalog do pracy i przejdź tam (mkdir lab1; cd lab1)
5. wyświetlić aktualną datę (date)
6. sformatować datę YYYY/MM/DD hh:mm:ss (date "+%Y/%m/%d %H:%M:%S")
7. wpisać sformatowaną datę do pliku data.txt (date "+fmt" > data.txt)
8. utworzyć plik vi tekst.txt, nauka obsługi vi
9. skleić pliki data.txt,tekst.txt do pliku razem.txt (cat)
10. utworzyć archiwum (tar) z wszystkich plików txt (tar cf arch.tar *.txt)
11. utworzyć nowy katalog i przejść tam (lab11)
12. odzyskać pliki z archiwum w katalogu lab11

Skrypty Bash

Każdy skrypt (program wykonywany przez interpreter) MUSI w pierwszej linii zawierać wskazanie (pełną ścieżkę) do właściwego interpretera.
W przypadku skryptu bash będzie to:
#!/bin/bash

Na początku skryptu należy umieścić weryfikację parametrów a w razie wykrycia błędu wyświetlić informację o sposobie użycia skryptu.

Argumenty skryptu (lub funkcji)
$0 – nazwa skryptu
$1..$9 – pierwsze 9 argumentów skryptu
$# – liczba argumentów
"$@" – wszystkie argumenty (ze spacja jako separatorem)
"$*" – wszystkie argumenty (z IFS jako separatorem)
shift [N] – przesuń argumenty ($#=$#-1,$1=$2,$2=$3....)
dalsze argumenty dostępne jako ${10}, ${11} …

$$ – identyfikator procesu bash
$? – kod zakończenia wywołanego programu

$(command) == `command`
$(< file) == $(cat file) == `cat file`

Instrukcje basha: więcej można znaleźć w manualu (**man bash**)
for *name [in word] ;* ***do*** *list ;* ***done*** – pętla wyliczeniowa
for *((expr1 ; expr2 ; expr3)) ;* ***do*** *list ;* ***done*** – pętla jak w C
select *name [in word] ;* ***do*** *list ;* ***done*** – wyświetla numerowaną listę wyboru
case *word in [[(] pattern [| pattern] ...) list ;;] ...* ***esac*** – wybór instrukcji wg wzorca
if list; **then** list; [**elif** list; **then** list;] ... [**else** list;] **fi** – instrukcja warunkowa
while list; **do** list; **done** – pętla dopóki
until list; **do** list; **done** – jw
[**function**] name () { list; } – definicja funkcji

Równoważne instrukcje warunkowa
[-e plik] && echo "istnieje" || echo "nie istnieje"
if [-e plik]; then echo "istnieje"; else echo "nie istnieje"; fi
Zaawansowane operacje na zmiennych
${NAME:OFFSET:LENGTH} pobiera fragment tekstu lub tablicy
Zastosowanie wzorców
${NAME#pattern} usuwa z początku najkrótszy fragment pasujący do wzorca
${NAME##pattern} usuwa z początku najdłuższy fragment pasujący do wzorca
${NAME%pattern} usuwa z końca najkrótszy fragment pasujący do wzorca
${NAME%%pattern} usuwa z końca najdłuższy fragment pasujący do wzorca
${NAME/pattern/string} zastępuje pierwszy pasujący fragment do wzorca
${NAME//pattern/string} zastępuje każdy pasujący fragment do wzorca
uwaga: dopasowanie na takich zasadach jak dla plików
Zastosowanie wartości domyślnych
${NAME:-word} jeśli NAME jest niezdefiniowane wynikiem jest słowo (NAME bez zmiany)
${NAME:=word} jeśli NAME jest niezdefiniowane wynikiem jest słowo (oraz NAME=word)
${NAME:?word} jeśli NAME jest niezdefiniowane wynikiem jest słowo (oraz wypisane na stderr)
${NAME:+word} jeśli NAME jest zdefiniowane wynikiem jest słowo

Użyteczne linki:
http://linuxconfig.org/Bash_scripting_Tutorial

Zadania (skrypty do napisania) do końca zajęć

1. **baskup.sh** backup katalogu do tgz
   ```
   fn=`pwd|tr "/" "_"`
   tar --exclude "*.tgz" -czf ~/${fn}_$(date +%Y%m%d).tgz .
   ```
2. **args.sh** wypisuje liczbę argumentów i każdy arg w osobnej linii
   ```
   echo Liczba arg: $#
   for a; do echo $a; done  #metoda 1
   for a in "$@"; do echo $a; done  #metoda 2
   while [ $# -gt 0 ]; do echo $1;shift; done  #metoda 3
   ```
3. **argsort.sh** sortuje argumenty
   ```
   (for a; do echo $a; done)|sort
   ```
4. **sum.sh** wyświetla sumę argumentów
   ```
   s=0;
   for a; do s=$((s+a)); done
   echo suma: $s
   ```
5. **fzero.sh** znajdź pliki zerowej długości w katalogu
   ```
   find ${1:-"."} -type f -a -size 0c
   ```
6. **fmax.sh** znajdź największe pliki w katalogu
   ```
   find ${1:-"."} -type f -printf '%s %p\n'|sort -n|tail -10
   find ${1:-"."} -type f -a ! -type d -printf '%s %p\n'|sort -n|tail -10
   ```
7. **category.sh** wyświetla klasyfikuje pliki w katalogu o nazwie podanej w argumencie
   (czy istnieje,czy plik,czy katalog,czy można czytać, czy wykonywalny)
   ```
   find ${1:-"."}  -name "$2"|while read fn; do
     s=""
     if [ -x "$fn" ]; then s="$s, exec"
     [ -x "$fn" ] && s="$s, exec"
   done
   ```
8. **replace.sh** zamień frazy w pliku
   ```
   src=$3
   dst=$4
   find ${1:-"."}  -name "$2"|while read fn; do
     sed -e "s/src/dst/g" "$fn" > /tmp/repl.tmp
     mv /tmp/repl.tmp "$fn"
   done
   ```