Elektrotehnički fakultet
Univerziteta u Beogradu
Katedra za računarsku tehniku i informatiku

Praktikum iz Operativnih sistema

- rešenja za 2005. godinu -

Verzija: 24.9.2007. 1:13

1. **Napisati pet shell komandi i objasniti čemu služe. Za svaku od komandi, navesti po dve opcije i objasniti šta se njima postiže.**

   Napomena: jasno je da rešenje koje je ovde dato nije jedino moguće, imajući u vidu postavku pitanja – ovo rešenje, pre svega, služi da studentima ukaže na stil odgovaranja na pitanja koja se tiču shell komandi.

`alias`

Prikazuje postojeće i postavlja nove smene simbola (engl. alias).

Bez parametara, ispisaće postojeće smene. Ako se koristi oblik **`alias IME=VREDNOST`**, postavlja nova smena, tako da se svaki put kad korisnik otkuca simbol **`IME`**, to interpretira kao **`VREDNOST`**.

`cd`

Menja tekući direktorijum u onaj koji je određen argumentom.

Ako se pozove bez argumenata, vraća korisnika u korisnikov **`home`** direktorijum. Poziv sa argumentom **`-`** vraća korisnika u prethodni direktorijum.

`pwd`

Ispisuje tekući direktorijum.

Opcija **`-P`** će ispisati stvarni, fizički tekući direktorijum. Opcija **`-L`** će ispisati simboličku vezu (engl. symbolic link) preko koga je korisnik stigao do tekućeg direktorijuma.

`jobs`

Ispisuje spisak trenutno aktivnih poslova koje je korisnik pokrenuo.

Opcija **`-r`** ograničava ispis samo na poslove koji se trenutno izvršavaju ("**`Running`**"). Opcija **`-s`** ograničava ispis samo na poslove koji se trenutno izvršavaju ("**`Stopped`**"). Opcija **`-l`** ispisuje i **`PID`** za svaki od poslova.

`logout`

Završava tekuću shell sesiju.

2. **Opisati prava pristupa fajlovima kod Linux OS.**

   Postoje tri vrste pristupa do fajlova/direktorijuma: čitanje (**`Read`**), pisanje (**`Write`**) i izvršavanje (**`eXecute`**). Za svaku od ovih vrsta koristi se po jedan bit (**`1`** je aktivna vrednost bita). Bitovi su označeni sa **`R`**, **`W`** i **`X`**. Postoje tri paketa od ova tri bita, svaki za određenu vrstu korisnika: za korisnika koji je vlasnik fajla (**`owner`**), za grupu koja je grupni vlasnik fajla (**`group owner`**) i za sve ostale korisnike sistema.

```
drwxrw-r-- 12 andrija pos 4096 2007-06-04 02:03 hello.c
```

značи da fajlu **`hello.c`** korisnik **`andrija`** može pristupiti na sva tri načina, svi korisnici iz grupe **`pos`** imaju pristup za čitanje i za upis, dok ostali korisnici mogu samo čitati taj fajl.

3. **Navesti komande koje služe za manipulaciju pravima pristupa.**
4. **Detaljno objasniti komande `chmod`, `chown`, `chgrp`.**

   `chmod` menja prava pristupa do navedenog fajla/navedenih fajlova.

   `chown` i **`chgrp`** menjaju vlasnika i grupnog vlasnika navedenog fajla/navedenih fajlova.

5. **Detaljno objasniti komandu `chmod` i navesti sve načine promene prava pristupa pomoću ove komande.**

   chmod menja prava pristupa do navedenog fajla/navedenih fajlova. Postoje dva načina primene ove komande: simbolički i numerički (oktalni).

   Simbolički pristup podrazumeva navođenje imena vrste korisnika na koju se komanda odnosi, novog prava pristupa i imena bita koji treba promeniti. Vrsta korisnika se određuje sa **u** (**user**), **g** (**group**), **o** (**others**) i **a** (**all**). Nova prava pristupa se određuju sa **=**, **+** i **-**, dok se za ime bita navodi **r** (**read**), **w** (**write**) ili **e** (**executable**). Kod **+** i **-**, bitovi koji ne budu navedeni u komandi neće biti promenjeni. Ovako ustrojena uređena trojka se može ponoviti u istoj komandi. Primer: **chmod u+rwx,g+rw,o-wx hello.c** će na fajlu **hello.c** postaviti na **1** sva tri bita za vlasnika fajla, postaviće na **1** bitove za čitanje i pisanje za grupnog vlasnika fajla i postaviće na **0** bitove za pisanje i izvršavanje za sve ostale korisnike.

   Numerički ili oktalni mod podrazumeva da se bitovi prava pristupa posmatraju kao trocifreni oktalni broj. Prva oktalna cifra se odnosi na vlasnika, druga na grupnog vlasnika, treća na sve ostale korisnike. Unutar svake od cifara, najznačajniji bit je **r**, sledeći bit je **w**, najmanje značajan bit je **x**. Numerički mod menja vrednosti svih bitova za sve grupe. Primer: **chmod 764 hello.c** će postaviti prava pristupa za fajl **hello.c** tako da vlasnik ima sva prava pristupa, grupni vlasnik može ima pravo čitanja i pisanja, dok svi ostali korisnici imaju samo pravo čitanja.

6. **Šta je shell script i čemu služi?**

   Shell script je programski jezik za upravljanje sistemom (najčešće iz konzole). Služi da se postave uslovi za izvršavanje većeg broja komandi i da se navede redosled izvršavanja tih komandi.

7. **Koji atribut mora da bude postavljen na fajlu u kome se nalazi skript?**

   Mora biti postavljen atribut za izvršavanje (**X**). Postavlja se sa **chmod +x** ime_fajla, ako se želi da sve tri klase korisnika sistema (vlasnik, grupni vlasnik, ostali) mogu da izvrše dati skript.

8. **Šta je i čemu služi `pipe`?**

   pipe je kanal za komunikaciju između procesa. Ima sve odlike datoteke bez mogućnosti slučajnog pristupa.

9. **Šta je i čemu služi `socket`?**

   socket je jedan od mehanizama za međuprocesnu komunikaciju. Može se koristiti lokalno (UNIX Socket) ili u mreži (Internet Socket). U užem smislu to je jedan kraj u vezi između procesa.

10. **Koja je razlika između `pipe` i `socket`?**

    socket podržava značajno veći broj protokola i ne mora biti ograničen na jedan sistem.

11. **Prikazati kako se koristi `named pipe`.**

    Kreiranje **named pipe** pod imenom **namedpipe** se obavlja sledećom komandom:

```
mknod namedpipe p
```

    Proces koji želi da sluša može da otvori taj **named pipe**:

```
tail -F namedpipe
```

Proces koji želi da upisuje takođe koristi **pipe** kao običnu datoteku:

```
ls > namedpipe
```

12. Koristeći `named pipe`, prikazati kako se može pokrenuti komanda za listanje sadržaja direktorijuma u jednom komandnom prozoru, a da se rezultat te komande prikaže u drugom prozoru.

Videti pitanje 11.

13. Čemu služi sistemski poziv `fork()`?

Služi da pokrene još jednu kopiju procesa iz koga je pozvan. Proces stvoren na ovaj način će imati svoj **PID** i zaseban adresni prostor. Njegov **PPID** će biti isti kao **PID** procesa koji je pozvao **`fork()`**.

14. Napisati deo koda, kojim se ilustruje korišćenje sistemskog poziva `fork()`.

```
/*
** fork1.c -- demonstrates usage of fork() and wait()
*/

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
     pid_t pid;
     int rv;

     switch(pid = fork()) {
     case -1:
          perror("fork");  /* something went wrong */
          exit(1);         /* parent exits */

     case 0:
          printf(" CHILD: This is the child process!\n");
          printf(" CHILD: My PID is %d\n", getpid());
          printf(" CHILD: My parent's PID is %d\n", getppid());
          printf(" CHILD: Enter my exit status (make it small): ");
          scanf(" %d", &rv);
          printf(" CHILD: I'm outta here!\n");
          exit(rv);

     default:
          printf("PARENT: This is the parent process!\n");
          printf("PARENT: My PID is %d\n", getpid());
          printf("PARENT: My child's PID is %d\n", pid);
          printf("PARENT: I'm now waiting for my child to exit()...\n");
          wait(&rv);
          printf("PARENT: My child's exit status is: %d\n",
WEXITSTATUS(rv));
          printf("PARENT: I'm outta here!\n");
     }
```

```
     return 0;
}
```

15. Šta je demonski proces (`daemon`)?

Demonski proces je proces koji je napravljen da radi u pozadini i nema povezivanje sa terminalom (nema pristup standardnim tokovima).

16. Kako se pravi `daemon`? Rešenje ilustrovati kodom.

Demonski proces se pravi tako što glavni program pozove `fork()` i u dečijem procesu zatvori sve standardne tokove. Glavni program se završava i njegovo dete postaje dete `init` procesa. Za sve ovo se može koristiti pomoćna funkcija `daemon`.

```
#include <unistd.h>
int main( int argc, char * argv[] )
{
     if ( -1 != daemon( 0, 0) )
          { /* Odavde nadalje se mozemo smatrati demon-om. */ }
     else
          { /* Desila se greska pri pokusaju da postanemo daemon. */ }
}
```

17. Za šta se koriste demonski procesi?

Najčešće se koriste za servise sistema, servere i slične namene.

18. Opisati TCP/IP protokol sa programerskog aspekta. Koji je osnovni razlog uvođenja ovog protokola.

Više nije u nastavnom programu IR2POS. Videti materijal za IR3RM.

19. Opisati rešenje sistema za prijem elektronske pošte koji je primenjen u EUNetu. Šta se dobija ovakvim rešenjem?

Više nije u nastavnom programu IR2POS. Videti materijal za IR3RM.

20. Prikazati sintaksu IF naredbe za `shell script`.

```
if komanda1
then
    komanda2
else
    komanda3
fi
```

21. Prikazati sintaksu FOR naredbe za `shell script`.

```
for i in *
do
    echo Naziv datoteke je: $i
done
```

22. Napisati `shell script`, koji će pozvati program test 100 puta, prvi put sa parametrom 1, pa sa 2, itd. Ime programa koji treba pozvati se skriptu prosleđuje kao parametar. Primer: `./pokreni test`.

```
#!/bin/bash
for i in `seq 1 100`
do
    $1 $i
done
```

23. Napisati na jeziku Java program koji se kači na 192.168.0.34:2345, šalje poruku "Zdravo" i zatim očekuje povratnu poruku, koju štampa na ekran.

```
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws IOException {
        Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;
        String hostname = "192.168.0.34";
        int port = 2345;
        try {
            echoSocket = new Socket(hostname, port);
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(
                    echoSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: " + port);
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                               + "the connection to: " + hostname);
            System.exit(1);
        }
        out.println("Zdravo");
        System.out.println(in.readLine());

        out.close();
        in.close();
        stdIn.close();
        echoSocket.close();
    }
}
```