Zadaci za vežbu

1. (jun 2014.) U datoteci *in.txt* se u prvom redu nalazi broj *n*, a zatim u svakom od narednih *n* redova putanja do direktorijuma. Maksimalna dužina putanje je 1023 karaktera. U svakom direktorijumu pročitati sve datoteke (samo direktno, ne rekurzivno). Napisati program *izvrsi_programe.c* koji izvršava svaku od tih *regularnih* datoteka koja ima prava izvršavanja za ostale, a modifikovana je u poslednjih 30 dana. Za svrhu testiranja može se napisati nekoliko jednostavnih skripti.

2. (septembar 2013.) Ime datoteke se prosleđuje kao argument komandne linije programa *sabiranje_pomocu_niti.c*. U datoteci su zadati brojevi n i k (n, k <= 100), a zatim i n*k brojeva koji predstavljaju članove niza a. Sabrati članove niza:

$(a_1+a_2+a_3+\ldots+a_k)+(a_{k+1}+a_{k+2}+a_{k+3}+\ldots+a_{2k})+\ldots$

Svaki zbir od k elemenata $a_{ik+1}+a_{ik+2}+a_{ik+3}+\ldots+a_{(i+1)k}$ treba da računa posebna nit i da ga doda ukupnom zbiru. Različite niti treba da se izvršavaju istovremeno. Ograničiti da promenljivu koja predstavlja ukupni zbir ne mogu da modifikuju dve niti istovremeno. Na primer, za ulaznu datoteku sadržaja:

20 2

1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

rezultat treba da bude 100.

3. Napisati program *r_d_r.c* koji kreira dete proces i demonstrira komunikaciju između roditeljskog procesa i deteta procesa pomoću *pipe* sistemskog poziva. Prvo roditelj šalje detetu poruku "r->d" a potom dete šalje roditelju poruku "d->r". Za obezbeđivanje da prvo roditelj pošalje poruku dovoljno je u detetu koristiti naredbu *sleep(3)*. Obezbediti da roditeljski proces čeka na dete proces. (Uputstvo: koristiti 2 poziva funkcije pipe). Primer pokretanja programa je:

./r_d_r

r->d

d->r

4. Napisati program *shell.c* koji predstavlja verziju shell-a. Program u petlji prikazuje na ekranu znak % i učitava komande korisnika. Po učitavanju komande kreira se dete proces koje izvršava tu komandu. Komanda se može sastojati od 1 ili 2 reči, tj. može imati maksimalno jedan argument. Primer pokretanja može biti:

./shell

% ls

Makefile shell shell.c shell.c~

% ls -l

total 24

-rw-r--r-- 1 user users 239 2011-06-04 10:53 Makefile

-rwxr-xr-x 1 user users 9833 2011-06-04 10:58 shell

-rwxr--r-- 1 user users 2062 2011-06-04 10:58 shell.c

-rwxr--r-- 1 user users 2056 2011-06-04 10:57 shell.c~

% pwd

/home/user /OS_zadaci/3

% ls /

backupDisk boot etc home lost+found media opt root srv tmp var

bin dev ftp lib mail mnt proc sbin sys usr

Dodati u program i poziv funkcije koja obrađuje SIGALRM tako što ispiše poruku da je signal primljen i nastavi dalje sa izvršavanjem.

5. Napisati program koji ispisuje sve fajlove (računajući i direktorijume) koji su modifikovani u datom broju proteklih minuta. Pretraga treba da se vrši samo u direktorijumu čija je putanja data prvim argumentom komandne linije i svim njegovim poddirektorijumima. Broj proteklih minuta dat je kao drugi argument komandne linije. Za dobijanje trenutnog vremena može se koristiti funkcija time. Ispis treba da se sastoji od putanje fajla i broja minuta od modifikacije. Jedno pokretanje može biti:

./zadnji_modifikovani .. 20

Fajl: ../Zadnji_modifikovani

Modifikovan pre 0 minuta

Fajl: ../Zadnji_modifikovani/zadnji_modifikovani

Modifikovan pre 1 minuta

Fajl: ../Zadnji_modifikovani/zadnji_modifikovani.c

Modifikovan pre 1 minuta

Fajl: ../Zadnji_modifikovani/Makefile

Modifikovan pre 0 minuta

6. U fajlu temperature.txt čuvaju se 5 najviših temperatura u svetu u junu 2014. godine. Napisati program koji kao argument komandne linije prima najveću temperaturu u nekom mestu za dan kada je

pokrenut. Potrebno je da ova temperatura zameni najmanju temperaturu u pomenutom fajlu ako je veća od nje. Prvo zaključati fajl za čitanje, pa ukoliko je potrebno izmeniti ga - zaključati samo deo koji se menja za pisanje. Pretpostavka je da su temperature dvocifreni pozitivni brojevi. Proveriti da li je zaključavanje uspelo tako što se proces po upisu uspavljuje na 10 sekundi, i potom pokrenuti drugu instancu programa.

Ukoliko je saržaj fajla 21 25 19 22 24, i program se pokrene sa

./najvise_temperature 23&

./najvise_temperature 26&

sadržaj fajla po završetku treba biti 26 25 23 22 24.

7. Napisati program koji predstavlja verziju shell-a. Program u petlji prikazuje na ekranu znak % i učitava komande korisnika. Po učitavnju komande kreira se dete proces koje izvršava tu komandu. Pretpostavka je da se komande sastoje samo od jedne reči. Posle izvršavanja svake komande, ispisuje se koliko je trajalo izvršavanje komande (početak izvršavanja je poziv fork funkcije a završetak je trenutak kada je roditelj dočekao dete). Na primer, ako se program pokrene sa

./trajanje_komandi

i potom unese komanda *top* i posle nekoliko sekundi pritisne taster q, onda bi trebalo da se

prikaže približno vreme koliko je ova komanda bila aktivna.

8. Napisati program za dobijanje informacije o tome koji procenat aktivnih procesa u datom trenutku na datom sistemu ima *init* proces za roditeljski proces. U svrhu izračunavanja tražene vrednosti, program treba da iskoristi eksterni, sistemski program *ps*, odn. preciznije činjenicu da komanda *ps ahxo ppid* štampa na standardni izlaz listu ID-ova roditeljskih procesa za svaki proces na sistemu; za prihvatanje ovog ispisa treba iskoristiti sistemski poziv *pipe*. Traženi procenat se štampa na ekran. (Savet:
koristiti funkcije *dup2* i *fdopen*).

Primer: jedna sesija bi mogla biti oblika:

./deca_init_procesa

14%

9. (septembar 2014.) Napisati program *broj_c_fajlove_nitima.c* koji računa ukupan broj .c fajlova u direktorijumu koji se navodi kao prvi argument komandne linije i njegovim poddirektorijumima. Analizu svakog poddirektorijuma obraditi u posebnoj niti. Jedna nit treba da analizira sve datoteke u direktorijumu koji se prosleđuje kao argument komandne linije i u slučaju .c fajla da poveća globalni brojač, a u slučaju direktorijuma da inicijalizuje novu nit koja će analizirati dati direktorijum. Očekivati da nijedan direktorijum nema više od 100 direktnih poddirektorijuma, i analiza svakog direktorijuma treba da sačeka da sve niti koje analiziraju poddirektorijume završe rad. Može se pretpostaviti da je maksimalna dužina putanje do fajla 1024 bajtova.

Hijerarhija direktorijuma:
test/

1.c
2.c
dir1/
	1.c
dir2/
	1.c
	test.txt

Pokretanje programa:
./broj_c_fajlove_nitima test/
4