eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA

2005/2006 Ikasturtea

Sistema Eragileak I

Konputagailuen Arkitektura eta Teknologia Saila
Dpto. de Arquitectura y Tecnología de Computadores

2006ko irailaren 6a

1 Ariketa [1 puntu]

a) Ondoko ***auskalo*** izeneko scripta emanik, azal ezazu zer egiten duen.

```
for i $*
do
     if [ -d $i ]
     then
          for j in $i/*
          do
               if [ -x $j ]
               then
                    echo $j
               fi
          done
     fi
done
```

b) ***errepikatuak_bilatu*** scripta programatu ondokoa egin dezan: uneko katalogoko fitxategi motako sarrera bakoitza lehen mailako azpikatalogo bakoitzean bilatuko du, eta aurkitzen badu, bere izena uneko katalogoko ***errepikatuak.txt*** fitxategiari gehituko dio.

2 Ariketa [2 puntu]

UNIXeko Sarrera/Irteerako sistema-deiak erabiliz (ikus azken orriko zerrenda), ondoko programa idatzi nahi dugu C lengoaian:

`zenbat_agerpen` `fitxategi patroi lerro_kop`

Programa honek hiru argumentu ditu, eta irteera estandarrean `fitxategi` testu-fitxategiko lehen `lerro_kop` lerroetatik zenbatetan `patroi` patroia agertzen den idatziko du.

Gerta daitezkeen erroreen kontrola egitea eskatzen da.

OHARRA. Fitxategiko lerroak luzera desberdinekoak izan daitezkeenez (1 eta MAX karaktere artekoa), lehendabizi `lerroa_irakurri()` funtzio lagungarria kodetzea gomendatzen da.

3 Ariketa [1,5 puntu]

`/etc/passwd` testu-fitxategiak UNIX sistema eragileko erabiltzaileei buruzko informazioak gordetzen ditu, erabiltzaile bakoitzeko lerro bakar batean. Besteak beste, erabiltzailearen kontuaren izena (*username*) gordetzen du. Honako programa egitea eskatzen da:

`zenbat_acaf_lehenen_artean` `lerro_kop`

Programa honek argumentu bakarra du (lerro kopuru bat), eta ***`/etc/passwd`*** fitxategiko lehen `lerro_kop` lerroetatik `acaf` patroia zenbat lerrotan agertzen den idatziko du irteera estandarrean, hau da, zenbat erabiltzaile diren `acaf` taldekoak.

Programa honen bi bertsio idaztea eskatzen da:

a) Lehen bertsio bat UNIXeko ***komando lerro bakar bat*** erabiliz (`lerro_kop` = 100).

b) Bigarren bertsio ***multiprogramatu*** bat C lengoaiaz, UNIXeko ***komandoak*** erabiliz (ikus bukaerako zerrenda) eta beharrezko iruditzen zaizkizun izenik gabeko buzoiez (*pipe*-ak) baliatuz.

(**OHARRA**. Ezin da “`exekutatu_programa`” funtzioa erabili.)

4 ariketa [2,5 puntu]

Lurralde bateko mugetan kontrol-mekanismo bat ezarri nahi da kontrolatzeko nor sartzen den eta nor ateratzen den. Horretarako, lurraldearen muga bakoitzean makina berezi bat ezarri da, zeinak hatz-marka digitalak hartu eta muga zeharkatu behar duen pertsona ea terrorista den edo ez aztertuko duen. Funtzionamendua hurrengoa litzateke: mugan dauden makinak `GENERATRIX` izeneko `UNIX` zerbitzari batera konektatuta daude, eta norbait muga zeharkatu behar duen bakoitzean, mugako makinak eskaera bat bidaltzen dio zerbitzariari, azken honek aztertuko duelarik ea terrorista den edo ez muga zeharkatu nahi duen pertsona.

Mugan dauden makinak soilik eskaera bat egin ahalko dute uneko, zeinak bezero prozesu bat sortuko duen eta honek eskaera `MAILTRIX` izena duen zerbitzariaren buzoian idatziko duen, eskaeraren formatua hurrengoa izanik:

```
struct eskaera {
       char pasaportea[20];      /* pasaporte-zenbakia */
       char hatz_marka[1024];    /* hatz-marka digitala */
       char buzoia[80];          /* bezeroaren buzoia */
   }
```

Aztertzeko ea muga zeharkatu behar duen pertsona terrorista den edo ez, hurrengo **funtzioa** erabiliko dugu:

```
int hatz_marka_aztertu(char *hatz_marka);
```

Funtzio honek parametro bezala pasatako hatz_marka aztertu, eta muga zeharkatu behar duen pertsona terrorista bada 1 itzultzen du, eta 0 bestela.

Guztiarekin eskatzen da, `hatz_marka_aztertu` funtzioaren emaitzaren arabera bezeroari erantzutea, mezuak hurrengo formatua izango duelarik:

```
struct erantzuna {
       int kodea;        /* 1: terrorista; 0: ez terrorista */
   }
```

`GENERATRIX` zerbitzariak gehienez **100** eskaera zerbitzatu ditzake uneoro modu konkurrentean, `hatz_marka_aztertu` funtzioak **CPU kontsumo altua** baitu.

Hurrengoa eskatzen da:

a) Marraz ezazu eskema bat sistema osatuko duten prozesu guztiak azalduz eta elkarren arteko komunikazio mekanismoak argi adieraziz.

b) Idatz ezazu bezeroaren, zerbitzariaren eta lagungarri ikusten dituzun beste prozesu guztien kodea.

c) Adierazi zer aldatu beharko litzatekeen, `hatz_marka_aztertu` funtzioa, funtzioa izan beharrean programa bat izango balitz. Azaldu aldaketak eta marraztu eskema berria. Ez da eredu berria kodetu behar, soilik marraztu eta azaldu.

Sistema-deiak:

```
int open(char *izena, int modua, int baimenak);
int close(int kanala);
int read(int kanala, char *buf, int luz);
int write(int kanala, char *buf, int luz);
int lseek(int kanala, long posiz, int nondik);
int link(char *iturburua, char *helburua);
int unlink(char *izena);
int stat(char *izena, struct stat *sbuf);
int fstat(int kanala, struct stat *sbuf);
int chdir(char *izena);
int fork();
int execlp(char *izena, char *arg0, char *arg1, ..., char *argn, NULL);
int execvp(char *izena, char *arg[]);
int wait(int *egoera);
void exit(int egoera);
unsigned long time(0);
int pipe(int *pfd);
int mknod(char *izena, int baimenak, int unitatea);
int dup(int kanala);
```

st_dev:	unitatearen zenbakia
st_ino:	inodo zenbakia
st_mode:	modua
st_nlink:	lotura kopurua
st_uid:	jabearen identifikatzailea
st_gid:	taldearen identifikatzailea
st_size:	tamaina bytetan
st_atime:	azken atzipenaren data
st_mtime:	azken aldaketaren data
st_ctme:	sortu zeneko data

stat egitura

Funtzioak:

```
DIR *opendir(char *path);
int closedir(DIR *dir);
struct dirent *readdir(DIR *dir)      /* dir->d_name: izena */
int mkfifo(char *izena, int baimenak);
char *strcpy(char *helburu, char *jatorri);
char *strcat(char *katea1, char *katea2);
int strcmp(char *katea1, char *katea2);
char *strstr(char *katea, char *azpikatea);
int atoi(char *katea);
```

UNIX-eko programa eta utilitateak:

chmod – fitxategien baimenak aldatzeko
echo – mezuak idazten ditu irteera estandarrean
grep – azpikateak bilatzen ditu fitxategietan
head – fitxategien lehen lerroak aukeratzen ditu
ps – prozesuei buruzko informazioa
sort – fitxategien lerroak alfabetikoki sailkatzen ditu
tail – fitxategien azken lerroak aukeratzen ditu
wc – lerroak, hitzak, eta karaktereak kontatzen ditu
who – sistemara konektatutako erabiltzaileei buruzko informazioa