

eman ta zabal zazu



Universidad Euskal Herriko
del País Vasco Unibertsitatea
INFORMATIKA FAKULTATEA

2006/2007 Ikasturtea

Sistema Eragileak I

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

2007ko irailaren 5a

1. ariketa [1 puntu]

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

```
if [ $# -gt 5 ]
then
    echo "Mezua_1"
    exit 1
fi
for i in $*
do
    for j in *
    do
        if [ -d $j ]
        then
            for k in $j/$i*
            do
                echo $k
                if [ -f $k ]
                then
                    echo "Mezua_2"
                else
                    echo "Mezua_3"
                fi
            done
        fi
    done
done
done
```

b) **exekutagarritasuna_egiaztatu** scripta programatu honakoa egin dezan: argumentu bezala fitxategi zerrenda bat jasoko du, eta uneko katalogoko azpikatalogoetan (maila bat jaitsiz) fitxategi bakoitza existitzen den aztertuko du. Horrela bada eta fitxategiak exekuzio baimena badu, bere izena pantailan idatziko du. Aldiz, fitxategia existitzen bada baina exekuzio baimenik ez badu, mezu bat idatziko du pantailan hori adieraziz.

Erabilpen adibidea: **exekutagarritasuna_egiaztatu f1.sh f2.txt f3.c**

2. ariketa [2 puntu]

a) UNIXeko Sarrera/Irteerako sistema-deiak erabiliz, hurrengo **funtzioa** kodetu behar da C lengoaian:

```
int patroia_barne_du(char *fitx_izena, char *patroia, int pat_luz);
```

Funtzio honek honakoa egingo du: bigarren parametro bezala pasatako patroia testu-katea (pat_luz luzera duena), lehen parametro bezala pasatako fitxategiko 100. posizioan badago, orduan funtzioak 1 bueltatuko du. Aldiz, fitxategiak ez badu patroia 100. posizioan edota fitxategiaren tamaina behar baino txikiagoa bada, orduan funtzioak 0 bueltatuko du. Azkenik, errorearen bat gertatzen bada, funtzioak -1 bueltatuko du.

b) Aurreko ataleko funtzioa eta UNIXeko katalogoak atzitzeko liburutegi errutinak erabiliz, hurrengo **programa** kodetu behar da C lengoaian:

```
fitxategi_kutsatuak_bilatu patroia
```

Programa honek uneko katalogoa errekorrituko du, eta fitxategi arrunt bakoitzeko ea argumentu bezala pasatako patroia 100. posizioan duen aztertuko du (hau da, ea patroiaz “kutsaturik” dagoen aztertuko du). Horrela bada, fitxategiaren izena irteera estandarrean idatziko du.

Honako funtzio lagungarria kodetutzat ematen da, zuzenean erabiltzeko:

```
int fitxategi_arrunta_da(int fitx_modua);
```

Funtzio honek parametro bezala fitxategi baten “modua” jasotzen du (stat sistema-deiari lotutako egituraren st_mode eremuari dagokiona), eta 1 bueltatzen du modu hura fitxategi arrunt bati badagokio, eta 0 ez badagokio.

3. ariketa [1,5 puntu]

UNIXeko **last** komandoak, sistemara egindako konexio guztien informazioa irteera estandarrean idazten du. Konexio bakoitzeko lerro bat idazten du irteera estandarrean, besteak beste erabiltzailearen izena, konexioaren unea eta iraupena idazten direlarik. Horrela, erabiltzaile bat behin baino gehiago konektatu bada sistemara, egindako konexio beste lerro idatziko dira.

Aurrekoa jakinik, hurrengo **programa** kodetu behar da:

```
konexio_kopurua erabiltzailea
```

Programa honek argumentu bakarra jasotzen du, erabiltzaile baten izena hain zuzen, eta erabiltzaile horrek egindako konexio kopurua (hau da, zenbaki bat) irteera estandarrean idatziko du.

Programa honen bi bertsio idaztea eskatzen da:

- Lehen bertsio bat UNIXeko **komando lerro bakar bat** erabiliz (suposatu erabiltzailea *acaf0123* dela).
- Bigarren bertsio **multiprogramatu** bat C lengoaiatz, UNIXeko **komandoak** erabiliz (ikus bukaerako zerrenda) eta beharrezko iruditzen zaizkizun izenik gabeko buzoiez (*pipe*-ak) baliatuz.

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

4. ariketa [2,5 puntu]

Musika bideoklipak deskargatzeko UNIX zerbitzari bat garatu nahi da, *DIF-TUBE* izenekoa. Bezeroek eskaerak luzatuko dizkiote zerbitzariari, eskaera bakoitzeko bideoklip bat deskargatu daitekeelarik. Zerbitzariak aldi berean gehienez 100 eskaera zerbitzatu ditzake. Eskaerak **MBX_ZERB** izeneko buzoian jasotzen ditu zerbitzariak. Eskaeren formatua honakoa da:

```
struct eskaera
{
    char bez_kodea[12];
    char pasahitza[8];
    char bideokliparen_izenburua[64];
    char bez_buzoia[20];    // bezeroaren buzoia
}
```

Zerbitzariak, eskaera bat jasotzen duenean, bezeroak bideoklipa deskargatzeko baimena duen edo ez egiaztatu beharko du, eta horretarako hurrengo funtzioa erabiliko da (funtzio honen exekuzioa oso azkarra da):

```
int baimena(char *bez_kodea, char *pasahitza);
```

Funtzio honek 0 bueltatzen du erabiltzailea baimendua dagoenean, eta -1 baimendua ez dagoenean. Bezeroak baimena badu, bideoklipa deskargatzeko honako funtzioa dugu, dagoeneko kodetuta:

```
void bideoklipa_deskargatu(char *bez_kodea, char *bideokliparen_izenburua);
```

Bideokliparen deskarga onartua izan bada, deskarga guztiz burutu ondoren bezeroari erantzun egin behar zaio. Erantzunak honako formatua du:

```
struct erantzuna
{
    char bez_kodea[12];
    char bideokliparen_izenburua[64];
    int eskaera_onartua;    // 1 onartua, 0 ukatua
}
```

Bideokliparen deskarga ez bada onartzen (bezeroak baimenik ez duelako), bezeroari ere erantzun egingo zaio ukapena adieraziz (eskaera_onartua = 0).

Hurrengoa eskatzen da:

- Marraz ezazu eskema bat sistema osatuko duten prozesu guztiak azalduz eta elkarren arteko komunikazio mekanismoak argi adieraziz.
- Idatz ezazu zerbitzariaren eta lagungarri ikusten dituzun beste prozesu guztien kodea (bezeroaren atala kodetuzat jotzen da, ez egin).
- Suposatu **bideoklipa_deskargatu** funtzio bat izan beharrean programa bat dela. Adierazi ezazu zer aldaketa egin behar den (a) ataleko eskeman, eta marraztu eskema berria (ez da ezer kodetu behar, soilik aldaketak 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 mkfifo(char *izena, int baimenak);
int dup(int kanala);
DIR *opendir(char *path);
int closedir(DIR *dir);
struct dirent *readdir(DIR *dir);
    struct dirent { long d_ino;      // Inode zenbakia
                   char *d_name;   // Izena
                   }

```

st_dev:	unitatearen zenbakia
st_ino:	inode 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_ctime:	sortu zeneko data

stat egitura

Funtzioak:

```

char *strcpy(char *helburu, char *jatorri);
char *strcat(char *katea1, char *katea2);
int strcmp(char *katea1, char *katea2);
int strlen(char *katea);
char *strstr(char *katea, char *azpikatea);
int atoi(char *katea);

```

UNIX-eko programa eta utilitateak:

```

echo - mezuak idazten ditu irteera estandarrean
grep - azpikateak bilatzen ditu fitxategietan
head - fitxategien lehen lerroak aukeratzen ditu
last - sistemara egindako konexioak pantailaratzen 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

```