eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA

2007/2008 ikasturtea

Sistema Eragileak I

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

2008ko irailak 3

Ariketa 1 [1 puntu]

1. Bedi honako scripta, ***auskako*** izenekoa.

```
if [ $# -ne 2 ]
then
   echo "Mezua_1"
fi
if [ -d $1 ]
then
   for i in $1/$2*
   do
      if [ -f $i ]
      then
         head -20 $i
      elif [ -d $i ]
      then
         ls $i
      else
         echo "Mezua_2: $i"
      fi
   done
else
   echo "Mezua_3"
fi
```

a) Zer egiten du ***auskalo*** scriptak?

b) Kodetu ezazu ***auskalo2*** scripta, hurrengoa egin dezan: argumentu bezala, patroi bat, fitxategi amaiera bat eta katalogo bat jasoko ditu. Hirugarren argumentu bezala pasatako katalogoan, bigarren argumentua bezala amaitzen diren fitxategi arrunt bakoitzeko, lehen argumentu bezala pasatako patroia fitxategiaren edukian badago, fitxategi-izena irteera estandarrean idatziko da.

Erabilpen adibidea:
`auskalo2 printf .c lab4`

Ariketa 2 [2 puntu]

Lurralde bateko Ogasuna Sailak **`zergadunak.dat`** izeneko fitxategia du UNIX makina batean. Fitxategi honetan lurraldeko zergadun guztiei buruzko informazioa gordetzen da. Zergadun bakoitzari dagokion informazioa honako eran gordetzen da: lehenengo byte-a zergadun motari dagokio; ondoren zenbaki oso bat gordetzen da, zergadunari aplikatu beharreko erretentzioa adierazten duena; azkenik, zergadunari buruzko beste informazio guztia gordetzen da, zati honek beti 10 Kbyte okupatzen duelarik.

1 byte	1 zenb. osoa	10 Kbyte		1 byte	1 zenb. osoa	10 Kbyte
Mota	% Erretentzioa	Beste informazio guztia	…	Mota	% Erretentzioa	Beste informazio guztia

Zerga arautegiaren aldaketa bat dela eta, '**P**' motako zergadunei (**P** = Pentsioduna) aplikatu beharreko erretentzioa %10a izatera pasa behar da. UNIXeko Sarrera/Irteerako sistema-deiak erabiliz, kodetu ezazu C lengoaian **`zergadunak.dat`** fitxategiko '**P**' motako zergadun guztiei erretentzioa %10era eguneratuko dien programa. Gerta daitezkeen erroreen kontrola egin behar da.

Ohar garrantzitsua. Datuen Babeserako Bulegoarekin segurtasun arazoak direla medio, ezinezkoa zaizu zergadun bakoitzari dagokion beste informazio guztia irakurtzea. Hau da, zure programak zergadun bakoitzaren mota eta (beharrezkoa bada) aplikatu beharreko erretentzioa atzituko ditu soilik.

Zenbaki oso bat fitxategi batean idazteko modua honakoa da (***`fd`*** idazketa moduan irekitako kanal bat da, eta ***`n`*** zenbaki oso bat): ***`write(fd, &n, sizeof(n));`***

Ariketa 3 [1,5 puntu]

UNIX sisteman exekutatzen ari den prozesu baten kopurua kalkulatuko duen programa eraiki nahi da, **`prozesuak_zenbatu`** izenekoa (adibidez, exekutatzen ari diren **`pico`** prozesuaren kopurua zenbatuko duena). Programa honek bi argumentu jasoko ditu: zenbatu nahi den prozesuaren izena, eta emaitza gordeko duen fitxategiaren izena. Programak pasatako fitxategian, sisteman exekutatzen ari diren prozesuaren kopurua idatziko du. Programaren erabilpen egitura honakoa da:

`prozesuak_zenbatu prozesuaren_izena fitxategia`

Programa honen bi bertsio egitea eskatzen zaizu:

a) Lehen bertsio bat UNIXeko ***komando lerro bakar bat*** erabiliz, suposatuz prozesuaren izena "**`bash`**" dela, eta fitxategiaren izena "**`KI_kopurua`**" dela.

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

Oharra. UNIX makina batean exekutatzen ari diren prozesu guztien zerrenda lortzeko, "**`ps -A`**" komandoa erabil daiteke.

Ariketa 4 [2,5 puntu]

Unibertsitate berritzaile bateko Errektorearen hauteskundea teknologia berriak erabiliz aurrera eramango da. Bozketa **HAUTESKUNDE_ZERB** izeneko zerbitzariak kontrolatuko du, eta bozketa aurrera eramateko, hauteskunde-mahai bakoitzean txartel-irakurgailu bat dago, zeinak sinadura-digitalaren bitartez hauteslearen autentifikazioa burutzen duen.

Zerbitzariak bi datu-base erabiltzen ditu: batetik bozketen emaitzak **EMAITZAK.DB** izeneko datu-basean gordeko dira; eta bestetik, hautesle guztien zerrenda duena (sinadura-digitala barne), **ZERRENDA.DB**. Hautesleak lehen aldiz bozkatzen badu, **ZERRENDA.DB** datu-basean bozketa egin dela idatziko da, eta bozketa **EMAITZAK.DB** datu-basean gehituko da; dena ondo joan dela erantzungo zaio dagokion hauteskunde-mahaiari. Aldiz, hautesleak aurretik bozkatu badu, zerbitzariak hauteskunde-mahaiari komunikatuko dio iruzurra egiten saiatu dela, bozketa ez da datu-basean gehitzen, eta Hauteskunde Batzorde Nagusiari iruzurraren nondik-norakoak jakinaraziko zaio (aurrerago azaltzen da nola egin).

Hauteskunde-mahai bakoitzean **bezero** prozesu bat egongo da. Bezeroak hurrengo egitura duen eskaera bidaliko dio zerbitzariari, hautesle batek bozketa egiten duen bakoitzean:

```
struct hautesle_bozketa {
     char sinadura[MAX];     /* hauteslearen sinadura-digitala */
     char mahaia[MAX];       /* hauteskunde-mahaia             */
     char taldea[MAX];       /* hauteslearen taldea            */
     char kandidatua[MAX];   /* kandidatua                     */
     char mahai_buzoia[MAX]; /* hauteskunde-mahai buzoia       */
}
```

Hautesleak, bozketa-eskaera egiterakoan **MBX_HAUTESKUNDE** buzoira bidaliko du (guzti honetaz bezeroa arduratzen da), eta zerbitzariak buzoi honen bitartez bozketa-eskaerak jasoko ditu. Bestetik, hauteskunde-mahai bakoitzak bere buzoia du, non bozketaren erantzuna jasoko duen.

Zerbitzariak, bozketen zenbatzea aurrera eramateko hurrengo hiru funtzioak erabiliko ditu. Jakin, datu-baseekin lan egiten duten funtzioek denbora asko behar dutela eta ondorioz, motelak direla:

a) **sinadura_egiaztatu**, zeinak egiaztatzen duen ea hautesleak aurretik bozkatu duen edo ez, eta aurretik bozkatu ez badu **ZERRENDA.DB** datu-basea eguneratzen du. Funtzioak **0** balioa itzultzen du dena ondo joan bada, **1** datuak okerrak badira, eta **2** iruzur egiten saiatu bada (aurretik bozkatu izan badu).

```
int sinadura_egiaztatu(char *sinadura, char *mahaia, char *taldea);
```

b) **bozkatu**, zeinak hauteslearen bozketa **EMAITZAK.DB** datu-basean gehituko duen.

```
void bozkatu(char *mahaia, char *taldea, char *kandidatua);
```

c) **iruzurra_jakinarazi**, zeinak Hauteskunde Batzorde Nagusiari iruzurraren berri emango dion, adieraziz nor izan den iruzurraren egilea.

```
void iruzurra_jakinarazi(char *sinadura, char *mahaia);
```

Zerbitzariak, bozketaren sinadura egiaztatu bezain pronto, hauteskunde-mahaira erantzun beharko dio (hauteskunde-mahaiaren buzoia erabiliz) adieraziz nola joan den egiaztapena (dena ondo, datu okerrak edo iruzurra). Erantzuna bidaltzeko hurrengo datu-egitura dugu:

```
struct erantzuna {
     char sinadura[MAX]; /* hauteslearen sinadura-digitala    */
     int emaitza;        /*    0 – dena ondo, bozketa onartua */
                         /*    1 – datu okerrak               */
                         /*    2 – iruzurra                   */
}
```

Beraz, iruzurra emango balitz (**sinadura_egiaztatu** funtzioak 2 itzultzen badu), hautesle-mahaira erantzuteaz gain Hauteskunde Batzorde Nagusiari iruzurraren berri emango zaio **iruzurra_jakinarazi** funtzioaren bitartez.

Aurreko guztiaz gain, sistemak gehienez **500** bozketa egin ditzazke aldi berean, datu-baseak atzitzerakoan arazorik ez izateko.

Hurrengoa eskatzen da:

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

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

c. Marraz ezazu eskema berri bat azalduz zeintzuk izango diren egin beharreko aldaketak, **sinadura_egiaztatu** funtzio bat izan beharrean programa bat izango balitz. Ez da ezer kodetu behar, soilik aldaketak azaldu eta eskema berria marraztu.

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);
     // posiz: 0 hasiera, 1 unekoa, 2 bukaera
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 mknod(char *izena, int baimenak, int unitatea);
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:	dispositiboaren zenbakia
st_ino:	inode zenbakia
st_mode:	baimenak
st_nlink:	lotura kopurua
st_uid:	jabearen identifikadorea
st_gid:	taldearen identifikadorea
st_size:	tamaina bytetan
st_atime:	azken atzipenaren data
st_mtime:	azken aldaketaren 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);
int sprintf(char *katea, char *formatua, ...);
```

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