eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA

2008/2009 ikasturtea

Sistema Eragileak I

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

2009ko irailak 2

ARIKETAK
1. ariketa [1 puntu]

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

```
if [ $# -lt 1 ]
then
   echo “1_mezua”
   exit 1
fi
for i in $*
do
   if [ -f $i ]
   then
      for j in *
      do
         if [ -d $j ]
         then
            if [ -f $j/$i ]
            then
               tail -3 $j/$i
            fi
         fi
      done
   fi
   if [ -d $i ]
   then
      echo “2_mezua: $i”
   fi
fi
```

a) Zer egiten du ***ez_dakit*** ?

b) Kodetu ezazu ***x_baimena_aldatu*** scripta, hurrengoa egin dezan: argumentu bezala fitxategien izenen bukaerak jasoko ditu, eta uneko katalogoko lehen mailako azpikatalogoetan bukaera horiek dituzten fitxategi arruntak bilatuko ditu. Aurkitutako fitxategi bakoitzari honakoa egingo dio: fitxategiak exekuzio baimena badu, kendu egingo dio; aldiz, exekuzio baimenik ez badu, orduan ipini egingo dio.

Erabilera adibidea:
x_baimena_aldatu sh exe txt

2. ariketa [2 puntu]

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

```
int katalogoa_da_eta_hutsik_dago(char *izena);
```

Funtzio honek 1 bueltatzen du parametro bezala pasatako izena katalogoa bada eta hutsik badago (hau da, soilik "." eta ".." sarrerak baditu). Beste edozein kasutan (ez da katalogoa, katalogoa izanik ez dago hutsik…), funtzioak 0 bueltatuko du.

b) Aurreko ataleko funtzioaz baliatuz, "garbitzaile" lanak egiten duen **programa** C lengoaian kodetzea eskatzen da. Programa honek argumentu bakar bezala katalogo baten izena jasoko du, eta katalogo horren azpikatalogo hutsak ezabatuko ditu.

OHARRA: Erroreen kontrola egitea eskatzen da.

3. ariketa [1,5 puntu]

Banketxe batek bere kutxazain automatikoen kudeaketarako aplikazio bat egitea eskatu digu. Aplikazio honek, **`mugimendu_handienak`** izenekoa, kreditu txartela batekin egindako 100 mugimenduen zerrenda irteera estandarrean idatziko du, non mugimendu hauek diru kopuru altuenekoak diren (kreditu txartela horrekin egindako mugimenduen artean, noski).

Beraz, **`mugimendu_handienak`** aplikazioak argumentu bakar bezala kreditu txartela baten zenbakia jasoko du (hau da, karaktere kate bat), honako eran erabiliko delarik:

```
mugimendu_handienak   kreditu_txartela_zenbakia
```

Kutxazainean egindako mugimendu guztien erregistroa **`mugimenduak.log`** izeneko testu fitxategian gordetzen da, honako egiturarekin (lerro bat mugimendu bakoitzeko):

```
diru_kopurua    kreditu_txartela_zenbakia    data_eta_ordua
```

Fitxategia data eta orduaren arabera ordenaturik dago (eta ez diru kopuruaren arabera). Diru kopurua beti 10 karaktereko testu-kate bat da (kopuru bat adierazteko 10 karaktere baino gutxiago behar badira, aurretik behar beste '**`0`**' jartzen dira).

Eskatutako aplikazioaren bi bertsio egitea eskatzen zaizu:

a) Lehen bertsio bat UNIXeko ***komando lerro bakar bat*** erabiliz (suposatu kreditu txartelaren zenbakia "**`1111222233334444`**" 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: Erroreen kontrola egitea eskatzen da.

4. ariketa [2,5 puntu]

Bezero/zerbitzari motako aplikazio bat garatzea eskatu digute, **radar_zerbitzu** izenekoa, lurralde batean kokatutako radarren datuak kudeatzeko. Radar bakoitzak, tokiko abiadura maximoa baino handiagoan doan auto bat detektatzean, autoari argazki bat ateratzen dio, fitxategi batean zerbitzariari bidaltzen diolarik, eta eskaera bat bidaltzen du zerbitzariaren buzoira (buzoiaren izena **MBX_RADAR** da), **struct detekzioa** egitura duena. Zerbitzariak bere buzoian jasotako eskaera bakoitza zerbitzatu behar du, tokiko abiadura maximoa %10ean gainditu baldin bada arau-haustearen notifikazioa autoaren jabeari bidali behar zaiolarik.

```
struct detekzioa {
   int id_radar;
   int abiadura_max;
   int abiadura_autoa;
   char argazki_fitx[MAX];
   ….; // Informazio gehiago
}
```

```
struct autoa {
   char matrikula[MAX];
   char emaila[MAX];
   ….; // Informazio gehiago
   ….; //
}
```

Eskaera bat zerbitzatzeak honako urratsak ditu:

1) Arau-haustearen notifikazioa autoaren jabeari bidali behar zaion ala ez egiaztatu (operazio hau oso azkarra da). 2) Notifikazioa bidali behar denean, autoaren argazkiaren fitxategia OCR bidez aztertu behar da, matrikula lortzeko (operazio motela). 3) Autoaren datuak lortu behar dira, trafiko poliziaren datu-basetik (operazio motela). 4) Autoa datu-basean aurkitzen bada, bere jabeari notifikazioa emailez bidaliko zaio. Ez bada aurkitzen, email bat bidaliko da Isunen Gestiorako Sailaren helbidera (isungest@trafikoa.net), beraiek arduratu daitezen isunaren ordainketaz.

Errendimendu arrazoiak direla eta, gehienez 200 eskaera zerbitzatu ahal izango dira aldi berean.

Autoaren matrikula lortzeko ***OCR_Argazki*** izeneko **funtzioa** dugu (kodetutzat hartzen da):

```
void OCR_Argazki(char *argazki_fitx, char *matrikula);
```

Autoaren argazkia prozesatzen du, eta **matrikula** parametroan prozesaketaren emaitza uzten du (hau da, autoaren matrikula).

Autoaren datuak trafiko datu-basetik lortzeko ***DB_Kontsulta*** izeneko **funtzioa** dugu (kodetutzat hartzen da):

```
int DB_Kontsulta(char *matrikula, struct autoa *datuak);
```

Trafiko datu-basea kontsultatzen du, 1 bueltatuz pasatako matrikula aurkitzen badu, eta 0 aurkitzen ez badu. Aurkituz gero, **datuak** parametroan autoaren datuak bueltatuko dira.

Azkenik, emailen bidalketarako ***EmailaBidali*** izeneko **funtzioa** dugu (kodetutzat hartzen da):

```
void EmailaBidali(char *emaila,
                  struct detekzioa *detek,
                  struct autoa *datuak);
```

Email bat bidaltzen du pasatako helbidera, arau-hausteari buruzko informazioekin eta autoaren datuekin.

Hurrengoa eskatzen da:

1. Marraz ezazu eskema bat sistema osatuko duten prozesu guztiak azalduz eta elkarren arteko komunikazio mekanismoak argi adieraziz.
2. Idatz ezazu zerbitzariaren eta lagungarri ikusten dituzun beste prozesu guztien kodea.

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 despl, 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
                   }
     struct stat {
          st_dev:    unitatearen zenbakia
          st_ino:    inodo zenbakia
          st_mode:   modua
          st_nlink:  lotura kopurua
          st_uid:    jabearen identifikatzailea
          st_gid:    erabiltzaile-taldearen identifikatzailea
          st_size:   tamaina bytetan
          st_atime:  azken atzipenaren data
          st_mtime:  azken aldaketaren data
          st_ctme:   sortu zeneko data
     };
```

POSIX boolear makroak mota konprobatzeko (`m` `st_mode` eremua da):

```
        S_ISLNK(m)     S_ISREG(m)     S_ISDIR(m)     S_ISFIFO(m)
```

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:

chmod – fitxategien baimenak aldatzeko
echo – mezuak idazten ditu irteera estandarrean
grep – azpikateak bilatzen ditu fitxategietan
head – fitxategien lehen lerroak aukeratzen ditu
sort – fitxategien lerroak alfabetikoki sailkatzen ditu
tail – fitxategien azken lerroak aukeratzen ditu
wc – lerroak, hitzak, eta karaktereak kontatzen ditu