Pràctica 2.1

Objectiu

L'objectiu d'aquesta pràctica és aprendre conceptes bàsics relacionats amb la programació amb múltiples fils d'execució (*multithreading*).

Grups de pràctiques

Els grups de pràctiques seran de dues persones.

Entorn de treball

Les implementacions s'han de poder executar sobre el servidor *bsd.labdeim.net (**bsd**)*

Lliuraments

El lliurament de la pràctica 2 (fitxers font) cal fer-lo a través de l'aplicació MOODLE. La data límit dels lliuraments en primera i segona convocatòries es poden consultar a l'enllaç 'Distribució de classes' a l'espai *Moodle* de l'assignatura.

Correcció

La correcció de la pràctica 2.1 i 2.2 es realitzarà, en primera convocatòria, durant la sessió del laboratori, mitjançant una entrevista amb un professor i tots els membres del grup de pràctiques. Les dates d'entrevista estan especificades en la planificació en el Moodle de l'assignatura. A més, **es realitzarà una verificació automàtica de còpies entre tots els codis presentats**. Si es detecta alguna còpia, tots els alumnes involucrats (els qui han copiat i els que s'han deixat copiar) tindran la pràctica suspesa i, per extensió, l'assignatura suspesa.

Caldrà lliurar un **informe escrit** on es documenti la pràctica de la manera habitual, és a dir: *Especificacions, Disseny, Implementació* i *Joc de proves*. En l'apartat de joc de proves NO s'han d'incloure "fotos de pantalla" del programa, ja que la impressió sobre paper dels resultats no demostra que la implementació presentada realment funcioni. El que cal fer és enumerar (redactar) tots els casos possibles que el programa és capaç de tractar. Les proves presentades i altres afegides pel professor, s'executaran al servidor ***bsd*** o en un dels ordinadors de l'aula durant, abans o després de l'entrevista. El professor realitzarà preguntes sobre el contingut de l'informe i el funcionament dels programes presentats. Cada membre del grup tindrà una nota individual, que dependrà del nivell de coneixements demostrat durant l'entrevista.

Enunciat

Es tracta d'implementar un rudimentari joc del "menjacocos", fent servir un terminal de caràcters ASCII a través de la llibreria *'curses'*. En pantalla es dibuixa un laberint amb una sèrie de punts (cocos), els quals han de ser "menjats" pel menjacocos. Aquest menjacocos es representarà amb el caràcter '0', i el mourà l'usuari amb les tecles 'w' (adalt), 's' (abaix), 'd' (dreta) i 'a' (esquerra). Al mateix temps hi haurà un conjunt de fantasmes, representats pels caràcters del '1' al '9', que intentaran capturar al menjacocos. Evidentment, es tracta de menjar tots els punts abans que algun fantasma atrapi al menjacocos.

En l'espai moodle de l'assignatura es proporciona una versió inicial del programa '`cocos0.c`' on només s'hi presenta un fantasma. Per tal d'aprendre programació *multithreading*, es proposa modificar aquesta versió inicial de forma que l'ordinador pugui moure més d'un fantasma simultàniament. La idea principal és que la funció que controla el moviment del fantasma s'adapti per a ser executar com un fil independent o *thread*. Així, des del programa principal només caldrà crear tants fils d'execució com fantasmes es vulguin (fins a 9).

Fases de la pràctica

Per tal de facilitar el disseny de la pràctica es proposen 3 fases:

0. Programa seqüencial (`'cocos0.c'`):

Versió inicial amb un sol fantasma. Aquesta versió es proporciona dins de l'espai moodle.

1. Creació de threads (`'cocos1.c'`):

Partint de l'exemple anterior, modificar les funcions que controlen el moviment del menjacocos i del fantasma perquè puguin actuar com a fils d'execució independents. Així el programa principal crearà un fil pel menjacocos i un fil per a cada un dels fantasmes.

2. Sincronització de threads (`'cocos2.c'`):

Completar la fase anterior de manera que es sincronitzi l'execució dels fils a l'hora d'accedir als recursos compartits (per exemple, l'entorn de dibuix). D'aquesta manera s'han de solucionar els problemes de visualització que presenta la versió anterior.

Fase 0.

Dins de l'espai moodle hi ha disponibles una sèrie de fitxers per a fer la pràctica 2.

Per tal de generar la visualització del joc es proporcionen una sèrie de rutines dins del fitxer '`winsuport.c`', que utilitzen una llibreria d'UNIX anomenada *curses*. Aquesta llibreria permet escriure caràcters ASCII en posicions específiques de la pantalla (fila, columna). El fitxer '`winsuport.h`' inclou les definicions de les funcions implementades, juntament amb la descripció del seu funcionament:

```
int  win_ini(int *fil, int *col, char creq, unsigned int inv);
void win_fi();
void win_escricar(int f, int c, char car, unsigned int invers);
char win_quincar(int f, int c);
void win_escristr(char *str);
int  win_gettec(void);
void win_retard(int ms);
int  win_carregatauler(char *nom_fitxer, int n_f, int n_c, char creq);
```

- Cal invocar la funció `win_ini` abans d'utilitzar qualsevol de les altres funcions, indicant el número de files i columnes requerits de la finestra de dibuix, així com el caràcter per emmarcar el camp de joc i si aquest caràcter s'ha de visualitzar en atribut invers o no invers (definicions `INVERS` o `NOINV`). Els paràmetres de files i columnes es passen per referència perquè, si a l'invocar a la funció valen zero, la finestra ocuparà tota la pantalla del terminal i les variables referenciades contindran el número de files i columnes obtingut per la funció d'inicialització. L'última fila de la finestra de dibuix no formarà part del camp de joc, ja que es reserva per escriure missatges.
- En acabar el programa, cal invocar sempre a la funció `win_fi`.
- Per escriure un caràcter en una fila i columna determinades cal utilitzar la funció `win_escricar`. El paràmetre `invers` permet ressaltar el caràcter amb els atributs de vídeo invertits. El codi ASCII servirà per identificar el tipus d'element del camp de joc (espai lliure, paret, mejacocos, fantasma, coco).
- La funció `win_quincar` permet consultar el codi ASCII d'una posició determinada de la finestra de dibuix.
- La funció `win_escristr` serveix per escriure un missatge en l'última línia de la finestra de dibuix.
- Per llegir una tecla es pot invocar la funció `win_gettec`. Es recomana utilitzar les tecles definides en el fitxer de capçalera '`winsuport.h`' per especificar el moviment del menjacocos.
- La funció `win_retard` permet aturar l'execució d'un procés o d'un fil durant el número de mil·lisegons especificat per paràmetre.
- Finalment, la funció `win_carregatauler` permet llegir un fitxer de text de `n_f` files per `n_c` columnes on hi haurà representat el dibuix del laberint (paret, espais en blanc i cocos). A més, la funció visualitzarà en invers tots els caràcters del fitxer que coincideixin amb `c_req` (paret). Cal limitar les dimensions del laberint a una fila menys que la finestra de dibuix definida amb `win_ini`, ja que l'última fila s'ha de reservar per missatges de text.

El programa d'exemple ‘cocos0.c’ utilitza les funcions anteriors per dibuixar el camp de joc i controlar el moviment del menjacocos i del fantasma. Per generar l'executable primer cal compilar el fitxer ‘winsuport.c’, per tal d'obtenir un fitxer objecte ‘winsuport.o’ que contindrà les instruccions en codi màquina de les rutines de dibuix:

```
$   gcc -c winsuport.c -o winsuport.o
```

La comanda anterior només cal invocar-la una vegada per cada màquina on s'hagin d'executar les rutines de dibuix. És a dir, cal generar un fitxer ‘winsuport.o’ per al LINUX i un altre per al *bsd*. Després ja podem generar el fitxer executable ‘cocos0’ corresponent al joc del menjacocos bàsic. La següent comanda s'encarrega de fer això a partir del fitxer font ‘cocos0.c’, el fitxer objecte anterior i de la llibreria ‘curses’:

```
$   gcc cocos0.c winsuport.o -o cocos0 -lcurses
```

Ara ja es pot executar el programa. Abans però, s'han de conèixer els arguments que cal passar-li. El primer argument consisteix en el nom d'un fitxer de text que conté tots els paràmetres de configuració del joc. El format d'aquest fitxer ha de ser el següent:

```
n_fil1 n_col fit_tauler creq
mc_f mc_c mc_d mc_r
f1_f f1_c f1_d f1_r
```

on 'n_fil1', 'n_col' són les dimensions de la finestra de dibuix (dimensions del laberint més una fila pels missatges de text) i "fit_tauler" és el nom d'un fitxer de text que contindrà el dibuix del laberint, típicament amb número de files igual a 'n_fil1'-1 i número de columnes igual a 'n_col'. Els caràcters d'aquest fitxer iguals a 'creq' es visualitzaran invertits per tal de representar la paret del laberint. Els paràmetres 'mc_f', 'mc_c' indiquen la posició inicial (fila, columna) del menjacocos, i el paràmetre 'mc_d' indica la seva direcció inicial de moviment (0 -> amunt, 1-> esquerra, 2-> avall, 3-> dreta) i mc_r es correspon a un percentatge del retard introduit per paràmetre en el joc. Els paràmetres 'f1_f', 'f1_c', 'f1_d' i ‘f1_r’ es corresponen a la mateixa informació pel fantasma 1. El programa verifica que la primera posició del menjacocos i del fantasma no coincideixi amb un bloc de paret del laberint. Un exemple de fitxer de configuració és el fitxer ‘joc11.txt’:

```
$ cat joc11.txt
8 15 lab1.txt +
1 1 3 1.0
5 7 3 2.0
```

Per veure el dibuix del laberint, cal visualitzar el contingut del fitxer ‘lab1.txt’, on ‘+’ representa un bloc de paret i ‘.’ representa un coco:

```
$  cat lab1.txt
+++++++++++++++
+...+...+..+..+
+.++++....+++.+
+......+....+.+
+.++++++.+....+
+.....+.....+.+
+++++++++++++++
```

També de manera opcional, es pot especificar un segon argument per indicar el retard de moviment del joc (en mil·lisegons). Si no s'especifica res, el valor per defecte d'aquest argument és 100 (1 dècima de segon). Cal dir que aquest retard serà el que multiplicarà al valor entrat amb els paràmetres del menjacocos i dels fantasmes, en el cas de l'exemple el menjacocos tenia un valor de 1.0 i el fantasma un valor de 2.0, per tant, el menjacocos es mourà el doble de ràpid que el fantasma.

La comanda per executar el programa amb el fitxer de configuració '`joc11.txt`' i 250 mil·lisegons de retard de moviment és la següent:

```
$   ./cocos0 joc11.txt 250
```

L’estructura bàsica del programa principal és la següent:

```
int main(int n_args, char *ll_args[])
{
  :
  rc = win_ini(&n_fil1,&n_col,'+',INVERS);     /* intenta crear taulell */
  if (rc == 0)          /* si aconsegueix accedir a l'entorn CURSES */
  {
    inicialitza_joc();
    p = 0; fi1 = 0; fi2 = 0;
    do                  /********** bucle principal del joc **********/
    {
      fi1 = mou_menjacocos();
      p++;
      if ((p%2)==0)           /* ralentitza fantasma a 2*retard */
         fi2 = mou_fantasma();
      win_retard(retard);
    } while (!fi1 && !fi2);
    win_fi();
  }
  :
}
```

Bàsicament, el programa inicialitza la finestra de dibuix i executa el bucle principal, el qual activa periòdicament les funcions per moure el menjacocos i el fantasma.
El programa acaba quan s’ha premut la tecla `RETURN` o quan s'acaben els cocos o quan el fantasma es menja al menjacocos.

Fase 1.

En aquesta fase cal modificar les funcions anteriors de moviment del menjacocos i dels fantasmes perquè puguin funcionar com a fils d'execució independents.

Les capçaleres de les noves funcions de moviment han de ser les següents:

```
void * mou_fantasma(void * index)
void * mou_menjacocos(void * null)
```

En el cas de `mou_fantasma`, el valor que es passa pel paràmetre `index` serà un enter que indicarà l'ordre de creació del fantasma (0 -> primer fantasma, 1 -> segon fantasma, etc.). Cada fantasma s'escriurà com un caràcter invertit amb un codi ASCII diferent ('1' pel primer, '2' pel segon, etc.) i tindrà un retard diferent en funció del seu paràmetre de configuració. Per a la funció `mou_menjacocos` el paràmetre `null` no contindrà cap informació.

Al contrari que en la fase 0, les noves funcions de moviment han d'executar-se de manera independent al bucle principal del programa (*main thread*). Això implica que cada funció ha d'implementar el seu propi bucle per generar el moviment dels objectes. Per finalitzar l'execució dels fils de moviment, es poden fer globals les variables `'fi1'` i `'fi2'`, que indiquen alguna condició de final de joc (cocos esgotats, captura del menjacocos, tecla `RETURN`).

Per controlar els múltiples fantasmes, s'han de convertir les variables globals de posició i direcció dels fantasmes en un vector de màxim 9 posicions. Lògicament, caldrà ampliar la definició del fitxer de configuració per tal d'incloure la posició i direcció de moviment inicial de tots els fantasmes. Concretament, els paràmetres de cada fantasma ocuparan una nova línia per cada fantasma en el fitxer de configuració. El nou format del fitxer de configuració és el següent (compatible amb el format anterior):

```
n_fil1 n_col fit_tauler creq
mc_f mc_c mc_d mc_r
f1_f f1_c f1_d f1_r
f2_f f2_c f2_d f2_r
f3_f f3_c f3_d f3_r
...
```

Així, el nombre de files del fitxer determinarà quants fantasmes es volen crear. A més, s'ha d'observar que sobre un mateix laberint (fitxers `'lab?.txt'`) es poden especificar diferents posicions de menjacocos i fantasmes (fitxers `'joc??.txt'`). Com a exemples de fitxers de configuració amb múltiples fantasmes es proporcionen '`joc22.txt`' i '`joc34.txt`':

```
$ cat joc34.txt
16 53 lab3.txt +
1 11 2 0,5
5 1 3 2,0
1 26 2 0,75
13 37 0 1,0
1 51 1 1,25
```

També s'ha de modificar la funció d'inicialitzar el joc per dibuixar la posició inicial de tots els fantasmes carregats. El fil d'execució principal del programa s'ha d'encarregar de crear tots els fils de moviment i passar a executar una tasca independent fins que es doni alguna condició de finalització.

En el nostre cas, la tasca independent consistirà en un control de temps de joc (`minuts:segons`) i mostrar-ho per la línia de missatges (última línia del camp de joc). Quan s'arribi a una condició de finalització del programa, el fil d'execució principal sortirà del bucle i passarà a esperar que tots els altres fils acabin la seva execució abans de destruir l'entorn de dibuix. L'última acció del fil principal serà la d'escriure el temps total de joc i els cocos que falten per menjar per la sortida estàndard, juntament amb els missatges de finalització proposats en la fase anterior.

El fitxer executable s'anomenarà 'coco`s1`'. Per a generar-lo, s'ha d'invocar la següent comanda, la qual crida al compilador de C passant-li la referència de la llibreria que conté el codi de les funcions per treballar amb multithreading ('`pthread`'):

```
$  gcc cocos1.c winsuport.o -o cocos1 -lcurses -lpthread
```

ATENCIÓ: la implementació de la fase 1 no realitza cap mena de sincronisme entre els fils. Per tant, és possible que apareguin errors ocasionals a causa de l'execució concurrent i descontrolada d'aquests fils. No passa res, el propòsit d'aquesta fase és veure que s'executen tots els fils requerits en el fitxer de configuració (fins a 9) i el menjacocos.

Fase 2.

En aquesta fase cal establir seccions crítiques que evitin els problemes de concurrència de la fase anterior.

Les seccions crítiques han de servir per impedir l'accés concurrent a determinats recursos per part dels diferents fils d'execució. En el nostre cas, cal establir seccions crítiques per accedir a l'entorn de joc, per usar les rutines de curses, i per accedir alguna variable global que no permeti els accessos concurrents.

El fitxer executable s'anomenarà `'cocos2'`. Per validar el seu funcionament, caldrà executar-lo sobre el *bsd*.

Per tal d'agilitzar el procés de desenvolupament de la pràctica, es recomana la utilització d'un fitxer de text anomenat '`Makefile`', el qual ha d'estar ubicat en el mateix directori que els fitxers font a compilar. Per permetre la compilació de la nova fase, es poden afegir les següents línies:

```
$ vi Makefile
cocos2 : cocos2.c winsuport.o winsuport.h
      gcc cocos2.c winsuport.o -o cocos2 -lcurses -lpthread

winsuport.o : winsuport.c winsuport.h
      gcc -c winsuport.c -o winsuport.o
```

D'aquesta manera s'estableixen les dependències entre els fitxers de sortida (en aquest cas `'cocos2'` i `'winsuport.o'`) i els fitxers font dels quals depenen, a més d'especificar la comanda que s'ha d'invocar en cas que la data de modificació d'algun dels fitxers font sigui posterior a la data de l'última modificació del fitxer a generar. Aquest control el realitza la comanda *`make`*. Per exemple, podem realitzar les següents peticions:

```
$   make winsuport.o
$   make cocos2
$   make
```

Si no s'especifica cap paràmetre, `make` verificarà les dependències del primer fitxer de sortida que trobi dins del fitxer '`Makefile`' del directori de treball actual.