Pràctica 2.2

Objectiu

L'objectiu d'aquesta pràctica és aprendre conceptes bàsics relacionats amb la programació amb múltiples processos i múltiples threads.

Grups de pràctiques

Els grups de pràctiques seran de dos o tres alumnes.

Entorn de treball

Les implementacions s'han de poder executar sobre un servidor *bsd*.

Lliuraments

La practica2 tindrà dos fases de lliuraments, la primera fase, és la que us donem en aquest document. Aquesta primera fase de la pràctica 2 (fitxer font) es lliurarà a través de l'aplicació MOODLE. La data límit de lliurament en primera convocatòria serà abans del laboratori de processos (L9).

Correcció

La correcció de la pràctica 2.2 es realitzarà, en primera convocatòria, durant la sessió del laboratori «entrevistes P2», mitjançant una entrevista amb un professor i tots els membres del grup de pràctiques. Les dates d'entrevista en segona convocatòria s'especificaran a la pàgina Moodle de l'assignatura, així com la manera de concertar hora. 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.

El dia de l'entrevista cal portar 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 "pantallazos" 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 *sol* o a un ordinador del deim1, durant 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.

Fase 3.

L'objectiu d'aquesta fase és aprendre conceptes bàsics relacionats amb la programació amb múltiples processos (multiprocés), per tant, es proposa modificar el joc desenvolupat a la fase 2 de forma que ara s'utilitzin processos en lloc de *threads*. Concretament, cada pilota serà controlada per un procés fill. El codi d'aquests processos fill haurà de residir en un fitxer executable diferent del fitxer executable del programa principal (també anomenat programa pare).
Les funcions bàsiques de creació de processos, més la gestió de zones de memòria compartida, s'expliquen a la sessió L7 dels laboratoris.
A més del fitxer font principal `'mur3.c'` (procés pare), caldrà crear un altre fitxer font `'pilota3.c'` que contindrà el codi que han d'executar els processos fill per controlar les pilotes.
Per accedir a pantalla es disposa d'un nou grup rutines definides a `'winsuport2.h'`. Aquestes rutines estan adaptades a l'ús d'una mateixa finestra des de diferents processos. El seu funcionament és diferent de l'anterior versió de `winsuport` per a *threads*. Es disposa d'un exemple d'utilització als fitxers `'multiproc2.c'` i `'mp_car2.c'`.
A grans trets, la utilització de `winsuport2` ha de ser el següent:

- **Procés pare (codi del main en el mur3.c)**:
  - invoca a `win_ini()`: la nova implementació de la funció retorna la mida en bytes que cal reservar per emmagatzemar el contingut del camp de joc.
  - crea una zona de memòria compartida de mida igual a la retornada per `win_ini()`.
  - invoca a `win_set()`: aquesta funció inicialitza el contingut de la finestra de dibuix i permet l'accés al procés pare.
  - crea el thread de la paleta, tal com estava a la fase 2, però ara no es podran usar mutex, ja que les seccions critiques les haurem de fer també entre diferents processos.
  - crea el primer procés fill (primera pilota) passant-li com argument la referència (identificador IPC) de la memòria compartida del camp de joc, a més de les dimensions (files, columnes) i altres informacions necessàries.
  - executa un bucle principal on, periòdicament (p. ex. cada 100 mil·lisegons) actualitza per pantalla el contingut del camp de joc, invocant la funció `win_update()`.
  - un cop acabada l'execució del programa (inclosos tots els processos), invoca la funció `win_fi()` i destrueix la zona de memòria del camp de joc i altres recursos compartits que s'hagin creat.
- **Processos fill**:
  - mapejen la referència de memòria compartida que conté el camp de joc, utilitzant la referència (identificador) que s'ha passat per argument des del procés pare.

- invoquen la funció `win_set()` amb l'adreça de la zona de memòria mapejada anteriorment i les dimensions (files, columnes) que s'han passat per argument.
- utilitzen totes les funcions d'escriptura i consulta del camp de joc `win_escricar()`, `win_escristr()` i `win_quincar()`.

Els processos fill, però, no poden fer servir la funció `win_gettec()`, la qual només és accessible al procés que ha inicialitzat l'entorn de CURSES (el procés pare).

El fitxer executable principal s'anomenarà `'mur3'` i el de les pilotes d'ordinador s'anomenarà `'pilota3'`. Per generar-los, s'han d'utilitzar les següents comandes:

```
$    gcc -Wall mur3.c winsuport2.o -o mur3 -lcurses -lpthread
$    gcc -Wall pilota3.c winsuport2.o -o pilota3 -lcurses
```

L'execució es farà de manera similar a la pràctica anterior, invocant l'executable `'mur3'` amb els paràmetres corresponents al fitxer de configuració del joc (una fila per les dimensions més una fila per la paleta més una fila per cada pilota) i un valor opcional de retard.

Fase 4.

En aquesta fase cal establir seccions crítiques que evitin els problemes de concurrència de la fase anterior. També cal implementar la funcionalitat de comunicació entre els diversos processos fill (i potser també amb el procés pare), per practicar amb el mecanisme de comunicació entre processos.
Per establir seccions crítiques cal utilitzar semàfors. I per la comunicació entre processos es realitzarà mitjançant bústies de missatges. Les funcions bàsiques de manipulació de semàfors i de bústies per sincronitzar i comunicar els processos s'expliquen a la sessió 8 dels laboratoris de FSO.

En aquest punt voldrem afegir dues característiques especials al joc:

1. Voldrem fer que quan una pilota surti per la porteria, enviï missatges a totes les altres pilotes amb el seu **identificador** i la seva velocitat vertical (coordenada y), per exemple: «3:0.9».
   La resta de processos pilota rebran aquesta informació, canviaran el **símbol** de la seva pilota **a pantalla** pel de la que ha sortit, mentre que per la velocitat vertical aplicaran el següent algorisme:
   - si la velocitat pròpia és menor[1] que la velocitat rebuda llavors agafa la velocitat rebuda incrementada un 25%, sense canviar de sentit.
   - si la velocitat pròpia és major[1] s'agafa la velocitat de la rebuda, alentint-se, sense canviar de sentit .

   Recordeu que la velocitat de la pilota ha d'estar sempre entre -1,0 i 1,0.

2. Voldrem que quan una pilota toqui els **blocs T**[2] a més a més de fer-lo desaparèixer inicii un temporitzador en 5 segons. Durant el temps que duri aquest temporitzador, les pilotes s'han de dibuixar en mode invertit i quan toquin els blocs WLLCHAR (###) rebotarà però esborrarà el caràcter amb el que ha xocat, destruint parcialment la muralla.
   Si mentre dura el temporitzador es torna a tocar un bloc T, s'incrementarà el temps que quedava en 5 segons més.

3. En la part inferior de la pantalla, a més del temps de partida, s'ha de anar decrementant el temporitzador anterior per a saber el temps que ens queda disponible.

4. Per a facilitar els jocs de proves, quan es crei una pilota, si la creadora és lenta (vel_f<0.5) la nova serà ràpida (vel_f=1.0) altrament la nova haurà de ser lenta (vel_f=0.1). La vel_c nova pot ser random.

1 En valor absolut
2 A l'inicialització de la pantalla, canvieu el símbol dels blocs A superiors del taulell de joc per una T.