Pràctica 2.2

Objectiu.

L'objectiu d'aquesta pràctica és aprendre conceptes bàsics relacionats amb la programació amb múltiples processos (multiprocés).

Grups de pràctiques.

Els grups de pràctiques seran de dues o tres persones.

Entorn de treball.

Les implementacions s'han de poder executar sobre l'entorn LINUX dels laboratoris del departament i sobre el servidor *bsd*.

Lliuraments.

El lliurament de la pràctica 2.2 (fitxers font) cal fer-lo a través de l'aplicació MOODLE. La data límit de lliurament en primera convocatòria la podeu consultar en l'enllaç 'Distribució de classes' a la pàgina web de l'assignatura.

Correcció.

La correcció de la pràctica 2.2, en primera convocatòria, es realitzarà mitjançant una entrevista amb un professor i tots els membres del grup de pràctiques. L'entrevista tindrà una durada aproximada de 30 minuts. En una segona convocatòria es farà conjuntament amb la pràctica 2.1 si s'escau. Les dates d'entrevista s'especificaran a la pàgina web de l'assignatura, així com la manera de concertar hora. Per presentar-se a l'avaluació de la pràctica 2.2 tant en primera com en segona convocatòria, cal haver lliurat les pràctiques dins dels períodes establerts. Com sempre, **es realitzarà una verificació automàtica de copies entre tots els programes 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*. A 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 *bsd* 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.

Enunciat.

Per tal d'aprendre i practicar programació multiprocés, es proposa modificar el joc desenvolupat a la pràctica 2.1 de forma que ara s'utilitzin processos en lloc de *threads* per al control dels oponents. El codi d'aquests processos haurà de residir en un fitxer executable diferent del fitxer executable del programa principal (també anomenat programa pare), que serà carregat pels processos creats pel programa pare.

A més, per practicar la comunicació entre processos mitjançant bústies de missatges, els diferents objectes del joc hauran d'interactuar entre ells. Concretament, els oponents hauran de comunicar-se entre ells en el cas de xocar (més endavant s'ofereix informació més detallada).

Fases.

Per tal de facilitar el disseny de la pràctica 2.2 es proposen 2 fases:

1. Creació de processos (`'tron3.c'`/`'oponent3.c'`):

Partint del codi de la pràctica 2.1, realitzeu les modificacions necessàries per convertir els *threads* que controlen els trons oponents en processos independents. El control de l'usuari continuarà com un *thread* dins del procés pare. Així el programa principal crearà un *thread* per l'usuari i varis processos per als oponents. En aquesta fase no es demana cap mena de sincronització ni comunicació entre els processos que intervenen en el joc (la sincronització entre *threads* no funciona entre processos).

2. Sincronització de processos (`'tron4.c'`/`'oponent4.c'`):

Completar la fase anterior de manera que l'execució dels processos es sincronitzi 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 fase anterior, els quals es fan patents quan s'intenta executar el programa sobre el servidor remot *bsd*. En aquesta fase també s'ha d'implementar la interacció entre els trons oponents.

Fase 1.

En aquesta fase cal adaptar el codi de la pràctica 2.1 per poder treballar amb varis processos per controlar els oponents i un procés principal (amb diferents fils d'execució) per al control del tron de l'usuari i del temps d'execució del programa. Les funcions bàsiques de creació de processos s'expliquen a la L7 dels laboratoris de FSO. A més del fitxer font principal `'tron3.c'` caldrà crear un altre fitxer font `'oponent3.c'` que contindrà el codi que han d'executar els processos fill per controlar el moviment d'aquests.

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 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**:

1) 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.
2) crea una zona de memòria compartida de mida igual a la retornada per `win_ini()`.
3) invoca a `win_set()`: aquesta funció inicialitza el contingut de la finestra de dibuix i permet l'accés al procés pare.
4) crea els processos fill passant-los 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.
5) 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()`.
6) un cop acabada l'execució del programa (tots els processos), invoca la funció `win_fi()` i destrueix la zona de memòria del camp de joc.

! **Processos fill**:

1) 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.
2) 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.
3) utilitzen totes les funcions d'escriptura i consulta del camp de joc `win_escricar()`, `win_escristr()` i `win_quincar()`.
4) Dibuixen i porten el control total del seu rastre, inclòs el primer caràcter, escriuen en el fitxer de resultats el número de caràcters que han pogut realitzar sense xocar.

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 `'tron3'` i el dels trons oponents s'anomenarà `'oponent3'`. Per generar-los, s'han d'utilitzar les següents comandes:

```
$  gcc tron3.c winsuport2.o -o tron3 -lcurses -lpthread
$  gcc oponent3.c winsuport2.o -o oponent3 -lcurses
```

L'execució es farà de manera similar a la pràctica anterior, invocant l'executable `'tron3'` amb el paràmetres corresponents al número de trons oponents, la variabilitat de gir dels trons oponents i un valor opcional de retard.

Fase 2.

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 oponents. Per establir seccions crítiques cal utilitzar semàfors. La comunicació entre els oponents 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 L8 dels laboratoris de FSO.

Per tal practicar la comunicació entre els trons (usuari o oponents), aquests s'hauran de comportar de la següent manera:

- En el xoc de un tron A contra un altre tron B, el tron A haurà d'enviar un missatge al tron B per dir-li que ha estat xocat. Llavors, el tron B començarà a dibuixar-se de forma **NO INVERSA** (cal tenir en compte que inicialment s'escriuen de forma **INVERSA**). El tron A es destruirà.

- El tron B escriurà de forma **NO INVERSA** durant 10 segons aproximadament i passat aquest temps tornarà al seu estat **INVERS**.

- En el cas que un tron C xoqui contra el rastre **NO INVERS** del tron B que ha estat xocat anteriorment, el tron C també enviarà un missatge al tron B per dir-li que ha estat xocat, però aquesta vegada el missatge indicarà al tron B que és ell qui ha de morir, mentre que el tron C continuarà el seu recorregut 'travessant' la paret del tron B en forma sense canviar a l'estat **INVERS**.

El fitxer executable principal s'anomenarà `'tron4'` i el que controla els oponents `'oponent4'`. Per validar el seu funcionament, caldrà executar-lo sobre l'entorn del servidor *bsd*. La utilització d'un fitxer `'Makefile'` us facilitarà el desenvolupament de la pràctica.