Pràctica 2: Fronton Fase 2

Fonaments de Sistemes Operatius

Equip
Marc Cabré Guinovart
Aleix Mariné Tena
Assignatura
Fonaments de Sistemes Operatius
Data
24 / Maig / 2016

Especificacions

Per a la pràctica 2 se'ns demanava la implementació d'un joc tipus "fronton" implementat en 4 fases:

1. **Fase 1:** Implementació aplicant threading per a poder manipular les diferents parts de la aplicació en paral·lel.
2. **Fase 2:** Implementació amb sincronització mitjançant l'ús de seccions crítiques per evitar que els diferents threads puguin entrar en confilcte al accedir a variables comuns.
3. **Fase 3:** Implementació aplicant processos. Es demanava que s'implementés el que ja teniem amb processos en lloc de threads.
4. **Fase 4:** Implementació amb memòria compartida, sincronitzant els processos amb semàfors i amb bústies per la comunicació entre pilotes.

En aquest document es detallen les **fases 3 i 4** tot i que es comentaran detalls de les fases anteriors que afectin a aquesta entrega.

Per a aquesta entrega es demanava:

- Execució en paral·lel de les pilotes mitjançant processos. Per a això se'ns demana que tinguem dos fitxers de codi:
    - Fronton4.c
    - Pilota.c

  La creació dels diferents processos (1 per pilota) es realitza amb un fork, aplicant després un execlp per tal de carregant el codi de la pilota al procès creat.
- Creació d'un bloc de memòria compartida que permeti accedir a la informació comú entre els processos.
- Creació de semàfors de processos per tal de poder assegurar la sincronització al escriure i llegir de pantalla i al fer el recompte de rebots amb les paletes.
- Creació de bústies de missatges per establir una comunicació entre pilotes per tal de modificar la velocitat entre dues pilotes que xoquin.

Disseny

Per al disseny de la aplicació s'ha mantingut el disseny original del punt de partida. A partir d'aquí s'han afegit els canvis requerits:

- Fitxer Fronton.c: Conté el codi que controla el fil principal de l'aplicació i també la gestió de l'entorn de comuncació amb la part gràfica de la consola. També s'ocupa de controlar si es dona algun event que pugui fer acabar el joc i notificar-ho a les pilotes si s'escau.
- Fitxer Pilota.c: Conté el codi que gestiona el moviment de la pilota així com també les busties per llegir els canvis d'estat que es puguin succesïr degut a la interacció amb altres pilotes.
- Makefile: S'ha partit de l'original afegint algunes comandes per fer la compilació de la aplicació més còmode.
- Fitxers de prova amb format plà per tal d'aplicar diferents casuístiques de de funcionament.

Decisions de disseny

1. Per a la implementació de les bústies s'ha decidit que la manera més òptima és que la lectura de la bustia (que és bloquejant) es faci en un thread para·lel de manera que no afecti a la execució de la funció que gestiona el moviment de la pilota.
2. En memòria compartida només s'han inclòs les següents variables:
    - Rebots
    - Fi2
    - Retwin

    La resta de variables s'ha considerat que com que son d'ús puntual i no han de ser modificades per tots els processos, no era necessari posar-les en memòria compartida i les hem passat per paràmetre a la inicialització dels processos.
3. S'han considerat dos semàfors:
    - Sem_rebots: per tal d'actualitzar el contador de rebots
    - Sem_fi2: per tal de poder llegir i escriure la variable de fi de joc per pilota a la porteria amb seguretat.

Implementació

Pseudo-codi Fronton4.c

```
Inicialitzacio_variables()
Comprovacio_parametres_entrada()
Llegir_Configuració(parametre_entrada_1)
Si(error_llegir_configuració)
     Exit
Fi si

inicilaitza_tauler _joc()
creació_thread_paleta()
creació_processos_pilotes()

Fer:
     Wait_retard //delay per permetre la visualització
     Actualitzacio_pantalla()
Mentre: NoFi i rebots>0

Imprimir_resultat_joc()
```

PseudoCodi Pilota4.c

Main

```
Inicialitzacio_variables()
Comprovacio_parametres_entrada()
Crea_thread_lectura_busties()
Fer:
     Llegir_velocitat()
     Si (rebot_vertical)
          Si(caracter_Actual no es ' ')
               Comrprovar_rebot()
               si(caracter_actual es '0')
                    Decrementar_num_rebots
               Fisi
               Si (caracter no es '+')
                    llegir_pilota_xoc()
                    enviar_missatge_pilota_xoc()
               Fisi
          desa_nova_posicio_i_velocitat()
          Fisi
```

```
      Fisi
      Si (rebot_horitzontal)
            Si(caracter_Actual no es ' ')
                  Comrprovar_rebot()
                  si(caracter_actual es '0')
                        Decrementar_num_rebots
                  Fisi
                  Si (caracter no es '+')
                        llegir_pilota_xoc()
                        enviar_missatge_pilota_xoc()
                  Fisi
            desa_nova_posicio_i_velocitat()
            Fisi
      Fisi
      Si (rebot_diagonal)
            Si(caracter_Actual no es ' ')
                  Comrprovar_rebot()
                  si(caracter_actual es '0')
                        Decrementar_num_rebots
                  Fisi
                  Si (caracter no es '+')
                        llegir_pilota_xoc()
                        enviar_missatge_pilota_xoc()
                  Fisi
            desa_nova_posicio_i_velocitat()
            Fisi
      Fisi
      escriu_caracter()
```

Thread Bústia

```
Mentre (no sortir)
      llegir_missatge
      Si (entrada no es e) //comprovar si és exit
            Sortir = cert
      Sino
            Llegir_parametres_rebuts()
            Enviar_parametres_propis()
            Actualitzar_parametres_propis_amb_rebuts()
      Fisi
Fi mentre
```

Joc de proves

Per tal de poder validar bé el codi implementat, des de la primera fase, s'han generat fitxers de prova seqüencials que s'han anat modificant per tal de comprovar que es complia la casuística especificada a l'enunciat de la pràctica.

Aquests fitxers s'ha anomenta prova"n".txt i alguns d'ells s'han entregat amb la pràctica. Durant els processos de validació duts a terme s'han modificat aquests fitxers per tal de generar noves posicions i velocitats que permetessin comprovar, debugar i validar el correcte funcionament del codi.

Exemple de Fitxer:

```
20 30 9
8.5 13.5 -0.5 0.5
15.0 14.5 0.0 1.0
10.0 25.5 0.0 -0.5
5.0 25.0 1.0 0.0
```