Mariona Farré

Sistemes Operatius

Lab 1. Creació de Time Thieves (TT)

Objectius d'aquesta sessió

- Gestió d'arguments en la crida a un programa
- Gestió de processos: fork + exec
- Gestió de la finalització d'un procés: exit + wait
- Creació de programes a partir de diferents arxius

1. Gestió d'arguments

Es vol crear un TT que rebi dos valor, M i N, i comprovi que aquests son nombres enters més grans que 0. Altrament, escriurà un missatge informatiu i finalitzarà amb un -1.
A més, necessitem un replicant que rebi un valor, N, i la seva missió sigui comprovar que aquest es un nombre més gran que 0. Altrament, finalitzarà amb un -1.

a) Identifica l'enunciat amb els continguts de l'assignatura
   - •TT→ programa (codi) //en execució sera el procés cada vegada únics
   - • replicant → crear un nou procés "copiat" TT programa pare / replicat **programa fill** instrucció **fork**() procesos completament independents
   - •M, N → paràmetres/arguments entrades per teclat
   - •missatge informatiu → treure dades per pantalla **printf**.(--)/**write**(--)
   - • finalitzar amb -1 → instrucció per acabar procés **exit** (com finalitzar-ho (0,-1))

b) Crea un TT
c) Crea un replicant
d) Realitza les proves que creguis convenient per verificar el seu funcionament

2. Gestió de processos

Els TT han de tenir la capacitat de clonació i mutació canviant, així, la seva forma d'actuar per a ser com un replicant.
//poder fer que el replicant igualment que sigui "igual" al programa pare, s'haurà de mutar per aquest sigui diferent→ canviar-li el codi **execlp (-----)**, canvia el codi fill a un nou, sino anirà fent-se un bucle infinit (fork() →fork() →fork() →fork() →…..)

a) Identifica l'enunciat amb els continguts de l'assignatura
b) Fes que el TT crei M replicants que rebran un valor N cadascú.

//Crear M vegades de replicants entrat per pantalla que rebrà cada réplica un valor de N

c) Realitza les proves que creguis convenient per verificar el seu funcionament (diferents valors de M i N,...)

3. Gestió de la finalització

Els replicants finalitzen amb un 0 si la missió de comprovació del valor N, que els hi ha passat el TT, ha estat un èxit. La missió es considera un èxit quan N és més gran o igual a 0. Si la missió ha fracassat, el replicant ha de finalitzar amb un -1.
Per altra banda, el TT ha d'esperar que els seus replicants acabin la seva missió, recollint l'estat de finalització de cada replicant. En cas de que algun replicant no hi hagi tingut èxit en la seva missió, el TT escriurà un missatge informatiu.
//TT o programa pare necessita saber quan acabaran i com acabaran les seves rèpliques N>=0 si la missió ha sigut exitosa resultat==0 // si missió fracassa resultat==-1
//TT haurà d'esperar als seus replicants que retornin per saber el resultat final de missió
//**wait(**recull paràmetre tret pel **exit()), waitpid(--)** ← esperar un procés/replicant en concret

Utilitzar macros **wifexited-** si procés acaba be (amb l'exit) **wexitstatus** (recuperar valor exit)

a) Realitza un pseudocodi que expliqui clarament, en termes de l'assignatura l'enunciat.
b) Modifica el comportament del replicant
c) Modifica el comportament del TT

4. entrenant als replicants amb dummies

Els replicants han de tenir la capacitat de robar temps en blocs de N unitats i emmagatzemar-los en contenidors creat a la seva memòria.

//replicants agafa un bloc N i robar-lo

a) Identifica l'enunciat amb els continguts de l'assignatura
b) Modifica el replicant per a que tingui un contenidor a memòria «suficientment gran». Comprova que N es menor o igual que la mida del contenidor. En cas contrari el replicant finalitzarà amb un -1.

//Comprovar que N<= contenidor del replicant si missió complerta/ sino resultat -1
//roba chars i agafarà un array chars/ char buffer[ ] i descriure el tamany que tindran en cada contenidor

Per tal de entrenar als replicant abans d'atacar al Capitol, la resistència ha creat als dummies. El primer dummy que utilitzarem no fa res, simplement serveix per a que el replicant es familiaritzi amb ell.

//Nomeś la creació dels dummies que estan a dins d'una llibreria donada #include <..>

c) identifica l'enunciat amb els continguts de l'assignatura
d) Modifica el replicant per a que utilitzi al dummy de la següent forma:

```
(...)  // comprovacions inicials
dummy_ini( buffer, N );          //array creat, valor n
while (!final)
        final = dummy_cpt( buffer, N );
dummy_end();
```

//incloure aquest codi en el programa
→ tenir en el directori dummy.h i libdummy.a

```
gcc myprog.c -o myprog -L. -ldummy

gcc myprog.c -o myprog -L. -ldummy
gcc programaTT.c -o programaTT -L. -ldummy
gcc replicant.c -o replicant  -L. -ldummy
```

lab1:
TT.c

```
#include "dummy.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

void main(int argc, char *argv[]){
	if (argc !=3){
		printf("%s necesita dos parametres\n", argv[0]);
		}
	else{

		int M, N, pid, i, exitcode, stat;
		M = atoi(argv[1]);
		N = atoi(argv[2]);
		if (M>0 && N>0){
			for (i = 0; i < M; i++){
				pid = fork();
				if (pid==0){
					execlp("./replicant","./replicant",argv[2],(char*)0);
					}
			}
			for (i=0, exitcode=0; i<M; i++){
				wait(&stat);

				if (WIFEXITED(stat)){
					int8_t code;
					code = WEXITSTATUS(stat);
					exitcode = code;
					printf("Codigo [%d]\n", exitcode);
				}
				else{
					printf("Error en el exito del hijo\n");
					}

			}
			exit(0);
			}
		else{
			printf("%s necesita dos parametres majors que 0\n", argv[0]);
			exit(-1);
			}

	}
}
```

replicant.c

```
#include "dummy.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

void main(int argc, char *argv[]){
	if (argc !=2){
		printf("%s necesita un parametre\n", argv[0]);
		}
	else{

		int i, N, final;
		char buff[2000];

		N = atoi(argv[1]);
		if (N>0 && N<=2000){
			dummy_ini( buff, N );
			final = 0;
			while (!final){

				final = dummy_cpt( buff, N );
				}
			dummy_end();
			exit(0);
			}
		else{
			exit(-1);
			}
	}
}
```