PRACTICAS DE PROGRAMACIÓN CONCURRENTE 2020/2021. Sesión 1

Programación Concurrente

17 de septiembre de 2020

Procesos en Unix/C

Estudia el siguiente programa:

```
/***
procesos
***/

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

#define NUM_PROCESOS 5

int I = 0;

void codigo_del_proceso (int id) {
  int i;

  for (i = 0; i < 50; i++)
    printf("Proceso%d: i = %d, I = %d", id, i, I++);

  exit(id); // el id se almacena en los bits 8 al 15 antes de devolverlo al padre
}

int main() {
  int p;
  int id [NUM_PROCESOS] = {1,2,3,4,5};
  int pid;
  int salida;

  for (p = 0; p < NUM_PROCESOS; p++) {
    pid = fork ();
    if (pid == -1) {
      perror("Error al crear un proceso: ");
     exit(-1);
```

```
    }
    else if (pid == 0) // Codigo del hijo
      codigo_del_proceso(id[p]);
  }

  // Codigo del padre
  for (p = 0; p < NUM_PROCESOS; p++) {
    pid = wait(&salida);
    printf("Proceso %d con id = %x terminado", pid, salida >> 8);
  }
}
```

1. Comenta qué se espera que ocurra en cada porción de código y la salida.

2. Ahora edita y ejecuta un programa de ejemplo con dos procesos concurrentes que impriman en pantalla 1 y 2 respectivamente.

1. Transforma el código anterior para que definamos una única función a la que se le pase como parámetro el valor entero que se desea imprimir. Instancia a continuación dos procesos que ejecuten dicha función a la que le pasaremos como parámetro un 1 y un 2 respectivamente.

2. Implementa un programa que contenga una función imprimir que imprima un carácter cualquiera 5 veces. En el programa principal debemos crear 3 procesos concurrentes que impriman, utilizando la función imprimir, una ‘A‘, una ‘B‘ y una ‘C‘ respectivamente.

Hilos POSIX

Estudia el siguiente programa:

```
/**
  hilos
  Compilación: cc -o hilos hilos.c -lpthread
**/

#include <pthread.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define NUM_HILOS 5

int I = 0;

void *codigo_del_hilo (void *id){
   int i;
   for( i = 0; i < 50; i++)
      printf("Hilo %d: i = %d, I = %d\n", *(int *)id, i, I++);
```

```
   pthread_exit (id);
}

int main(){
   int h;
   pthread_t hilos[NUM_HILOS];
   int id[NUM_HILOS] = {1,2,3,4,5};
   int error;
   int *salida;

   for(h = 0; h < NUM_HILOS; h++){
      error = pthread_create( &hilos[h], NULL, codigo_del_hilo, &id[h]);
      if (error){
        fprintf (stderr, "Error: %d: %s\n", error, strerror (error));
        exit(-1);
      }
   }
   for(h =0; h < NUM_HILOS; h++){
      error = pthread_join(hilos[h], (void **)&salida);
      if (error)
         fprintf (stderr, "Error: %d: %s\n", error, strerror (error));
      else
         printf ("Hilo %d terminado\n", *salida);
   }
}
```

1. Comenta qué se espera que ocurra en cada porción de código y la salida. Comenta a continuación las diferencias más importantes entre este programa y el equivalente con procesos de la sesión 1.

2. Implementa un programa que contenga una función imprimir que imprima un carácter `n` veces. A la función se le debe pasar como parámetro una estructura en la que deben ir encapsulado el carácter y `n`. En el programa principal debemos crear 3 hilos concurrentes que impriman una ‘A‘ 50 veces, una ‘B‘ 100 veces y una ‘C‘ 150 veces respectivamente.

Entrega

- Adjunta a una tutoría en UA-Cloud hasta el dia 02 de octubre.