Sistemas Operativos I 2024

Trabajo Práctico 4 - threads y procesos

Objetivos

- Analizar diferencias entre **un único proceso, varios procesos, y threads**
- **API de C de llamadas al sistema** y funciones de manera general.

Referencias

[1] Tanenbaum, Bos – Modern Operating Systems - Prentice Hall; 4 edition (March 10, 2014) - ISBN-10: 013359162X
[2] Douglas Comer - Operating System Design - The Xinu Approach. CRC Press, 2015. ISBN : 9781498712439
[3] Silberschatz, Galvin, Gagne - Operating Systems Concepts - John Wiley & Sons; 10 edition (2018) – ISBN 978-1-119-32091-3

Software y Hardware

Este trabajo práctico se realiza en su totalidad en UNIX/Linux.

Ejercicio 1. *Programa para convertir una imagen a escala de grises, en PC UNIX/Linux.*

a. ¿Qué significa el acrónimo gcc?. ¿Desde cuándo está el proyecto?. ¿Cuál es su objetivo?
b. ¿Qué es POSIX?
c. Utilice el siguiente programa en lenguaje C:
   http://se.fi.uncoma.edu.ar/so/tp/convertir.tar.gz
   Observe la imagen input.bmp que se encuentra en la carpeta convertir/. Compilar y ejecutar el programa convertir_a_gris.c en un sistema UNIX/Linux.
d. ¿Considera que este programa podría ser portado fácilmente a MAC OS? ¿O a FreeBSD?. Si/no, porqué. Justifique su respuesta.

Ejercicio 2. *Sistema concurrente/paralelo compuesto de **varios procesos** en PC UNIX/Linux.*

a. Desarrolle un programa en C que genere tres procesos hijos. El padre debe esperar a que los hijos finalicen.
b. Divida el programa convertir_a_gris.c de tal manera que cada uno de los 3 hijos del programa del inciso a. procese una sección de la imagen y la convierta en gris. Por ejemplo, el primer hijo puede procesar las primeras 354 filas (la imagen original tiene 1063 filas de píxeles), el segundo hijo las siguientes 354 filas, y el último hijo las filas restantes. Luego que los hijos finalizan el padre debe escribir la nueva imagen en escala de grises (que se encuentra en memoria) a disco.
c. En el programa, ¿Cuáles llamadas a funciones son funciones de la biblioteca de C qué son simplemente útiles de manera general y cuales son funciones de la biblioteca de C que realizan llamadas al sistema?

Ejercicio 3. *Sistema concurrente/paralelo compuesto de **varios threads en un mismo proceso** en PC UNIX/Linux.*

a. Ejecute el programa debajo en un sistema GNU/Linux (compilar con la opción -lpthread).
b. Complete el programa para que cada uno de los threads realice la conversión a gris de una parte de la imagen, similar al Ejercicio 2.

```c
/* Ejemplo con threads en Linux. Compilar con: gcc -o p p.c -lpthread */

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

int total = 0;

void *thread(void * i)
{
        int n = *((int *) i);

        printf("Este es el thread nro: %i \n", n);

}


int main()
{
        int i0=0;
        int i1=1;
        int i2=2;
        pthread_t tid[3];

        /* Create independent threads each of which will execute function */
        pthread_create( &tid[0], NULL, thread, (void *) &i0);
        pthread_create( &tid[1], NULL, thread, (void *) &i1);
        pthread_create( &tid[0], NULL, thread, (void *) &i2);

        /* Wait till threads are complete before main continues.  */
        for (int i = 0; i <= 2; i++)
                pthread_join(tid[i], NULL);


        return 0;
}
```

Ejercicio 4. Responda

a. Utilice el commando **time** para medir los tiempos de los 3 programas que convierten_a_gris (Ejercicio 1, 2 y 3).
b. ¿Cuál versión fue la que tomó menos tiempo?. ¿Existe gran diferencia de ganancia entre hacerlo con procesos que con threads?. ¿Para qué tipo de programas puede que la ganancia sea mejor con threads?. Analice cuál de los mecanismos (un único proceso, varios procesos o varios threads) debería ser el más eficiente en cuanto a rendimiento (menor tiempo de ejecución).
c. ¿Por qué no existe en el sistema operativo XINU una llamada al sistema específica para crear threads o hilos?.