Trabajo Práctico N 2: Threads

1. Enumere las razones por las que un cambio de contexto entre hilos puede ser más barato que un cambio de contexto entre procesos (ventajas de los threads sobre los procesos).

2. Considere un servidor de archivos ejecutando como un proceso de un solo hilo en una máquina donde existen otros procesos ejecutándose. Para mejorar la performance, se desea agregar a dicho proceso soporte multihilo para tratar cada petición de archivo por separado. Se decide utilizar threads de nivel de usuario. Sin embargo, la performance del proceso no mejora. ¿Cuáles pueden ser la(s) razones(s) para que esto ocurra? Justifique.

3. Suponga un programa que crea n threads, donde cada thread posee una variable interna propia que se incrementa e imprime por pantalla en un bucle indefinidamente. En un momento dado, ¿es posible que un thread haya hecho muchas más impresiones por pantalla que otro? ¿En qué casos esto podría darse? Justifique.

4. **[LAB]** Implemente el siguiente código, compile y ejecute. Determine qué realiza cada parte del código. Ejecute reiteradas veces.

   Nota: *para compilar incluir la librería pthread.h.*

   a) ¿Siempre terminan todos los threads?
   b) Descomente las instrucciones comentadas, compile y ejecute nuevamente. ¿Qué hace dicha instrucción? ¿qué efecto tiene en este código?

```
#include <stdlib.h>
#include <pthread.h>
#include <windows.h>

#define MAX 10

void *funcion1()
{
    printf("Thread: %d\n", pthread_self());
    pthread_exit(0);
}

int main()
{
    pthread_attr_t attr;
    pthread_t thid[MAX];
    pthread_attr_init(&attr);
```

```
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
    for(int i = 0; i < MAX; i++){
        pthread_create(&thid[i], &attr, funcion1, NULL);
    }
    /**
    printf("Start sleep\n");
    Sleep(10000);
    printf("End sleep\n");
    */
    return 0;
}
```

5. **[LAB]** Implemente el siguiente código, compile y ejecute. Determine qué realiza cada parte del código. Ejecute reiteradas veces.
   a) ¿Siempre terminan todos los threads?
   b) Comente las líneas con las instrucciones pthread_join() y vuelva a ejecutar reiteradas veces. Describa y justifique el comportamiento observado.

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

void *funcion(void *p)
{
    printf("Thread: %d\n", pthread_self());
    pthread_exit(0);
}

int main()
{
    pthread_t th1, th2;
    pthread_create(&th1, NULL, funcion, NULL);
    pthread_create(&th2, NULL, funcion, NULL);
    printf("El thread principal continua ejecutando: %d\n", pthread_self());
    pthread_join(th1,NULL);
    pthread_join(th2,NULL);
    return 0;
}
```

6. **[LAB]** "Hello World!" V2.0... ahora con Threads!
   Escribir el código C que escriba en pantalla n veces el mensaje: "Hola Threads!. Soy el thread [idThread] ejecutando por [enésima] vez", donde [idThread] es el identificador de thread asignado por el lenguaje C y [n-ésima] es la cantidad de veces que se imprimió el texto por pantalla. Para obtener este identificador se puede utilizar el siguiente código: **getpid()**
   El código C debe crear 10 threads y ejecutar cada uno de ellos 10.000 veces (n = 10,000). Describir brevemente el resultado de la ejecución.