Sistemas Operativos

Práctica 6. Administrador de procesos en Linux y Windows (2)

Prof. Jorge Cortés Galicia

Competencia.

El alumno aprende a familiarizarse con el administrador de procesos del sistema operativo Linux y Windows a través de la creación de procesos ligeros (hilos) para el desarrollo de aplicaciones concurrentes sencillas.

Desarrollo.

1. A través de la ayuda en línea que proporciona Linux, investigue el funcionamiento de las funciones: **pthread_create(), pthread_join(), pthread_self(), pthread_exit(), scandir(), stat().** Explique los argumentos y retorno de cada función.
2. Capture, compile y ejecute el programa de creación de un nuevo hilo en Linux. Observe su funcionamiento.

```
#include <stdio.h>
#include <pthread.h>
void *hilo(void *arg);
int main(void)
{
        pthread_t id_hilo;
        pthread_create(&id_hilo, NULL, (void*)hilo, NULL);
        pthread_join(id_hilo, NULL);
        return 0;
}
void *hilo(void *arg)
{
        printf("Hola mundo desde un hilo en Linux\n");
        return NULL;
}
```

3. Capture, compile y ejecute el siguiente programa de creación de hilos en Linux. Observe su funcionamiento.

```
#include <stdio.h>
#include <pthread.h>
void *hilo(void *arg);
int main(void)
{
        pthread_t id_hilo;
        char* mensaje="Hola a todos desde el hilo";
        int devolucion_hilo;
        pthread_create(&id_hilo,NULL,hilo,(void*)mensaje);
        pthread_join(id_hilo,(void*)&devolucion_hilo);
        printf("valor = %i\n",devolucion_hilo)
        return 0;
}

void *hilo(void *arg)
{
        char* men;
        int resultado_hilo=0;
        men=(char*)arg;
        printf("%s\n",men);
        resultado_hilo=100;
        pthread_exit((void*)resultado_hilo);
}
```

4. A través del sitio MSDN, investigue el funcionamiento de la llamada al sistema CreateThread().
5. Capture, compile y ejecute el siguiente programa de creación de hilos en Windows. Observe su funcionamiento.

```
#include <windows.h>
#include <stdio.h>
DWORD WINAPI funcionHilo(LPVOID lpParam);
typedef struct Informacion info;
struct Informacion
{
   int val_1;
   int val_2;
};

int main(void)
{
   DWORD idHilo;                  /* Identificador del Hilo */
   HANDLE manHilo;                /* Manejador del Hilo */
   info argumentos;
   argumentos.val_1=10;
   argumentos.val_2=100;

   // Creacion del hilo
   manHilo=CreateThread(NULL, 0, funcionHilo, &argumentos, 0, &idHilo);

   // Espera la finalización del hilo
   WaitForSingleObject(manHilo, INFINITE);

   printf("Valores al salir del Hilo: %i %i\n", argumentos.val_1, argumentos.val_2);

   // Cierre del manejador del hilo creado
   CloseHandle(manHilo);
   return 0;
}

DWORD WINAPI funcionHilo(LPVOID lpParam)
{
   info *datos=(info *)lpParam;
   printf("Valores al entrar al Hilo: %i %i\n", datos->val_1, datos->val_2);
   datos->val_1*=2;
   datos->val_2*=2;
   return 0;
}
```

6. Programe una aplicación (tanto en Linux como en Windows), que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará 20 hilos. A su vez cada uno de los 20 hilos creará 15 hilos más. A su vez cada uno de los 15 hilos creará 10 hilos más. Cada uno de los hilos creados imprimirá en pantalla “Práctica 6 Hilo Terminal” si se trata de un hilo terminal o los identificadores de los hilos creados si se trata de un proceso o hilo padre.
7. Programe la misma aplicación del punto 5 de la práctica 5 en su sección de Linux pero utilizando hilos (tanto en Linux como en Windows) en vez de procesos, obtenga el tiempo de ejecución del programa con hilos. Compare ambos programas en su tiempo de ejecución (el creado en la práctica 5 y el creado en esta práctica), y dé sus observaciones tanto de funcionamiento como de los tiempos de ejecución resultantes.