Sistemas Operativos

Práctica 5. Administrador de procesos en Linux y Windows (1)

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 nuevos procesos por copia exacta de código y/o por sustitución de código para el desarrollo de aplicaciones concurrentes sencillas.

Desarrollo.

Sección Linux:

1. Introduzca los siguientes comandos a través de la consola del sistema operativo Linux:
   ps
   ps –fea
   pstree 0
   ¿Qué información le proporcionan los comandos anteriores?
2. A través de la ayuda en línea que proporciona Linux, investigue para que se utilizan los comandos **ps,** y **pstree,** mencione las opciones que se pueden utilizar con dichos comandos. Además investigue el uso de las llamadas al sistema **fork(), execv(), getpid(), getppid(), waitpid()** y **wait()** en la ayuda en línea, mencione que otras funciones similares a execv() existen, reporte sus observaciones.
3. Capture, compile y ejecute los dos programas de creación de un nuevo proceso por copia exacta de código que a continuación se muestra. Observe su funcionamiento y experimente con el código.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(void)
{
 int id_proc;
 id_proc=fork();
 if (id_proc == 0)
 {
  printf("Soy el proceso hijo\n");
  exit(0);
 }
 else
 {
  printf("Soy el proceso padre\n");
  exit(0);
 }
}
```

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(void)
{
 int id_proc;
 id_proc=fork();
 if (id_proc == 0)
 {
  printf("Soy el proceso hijo\n");
 }
 else
 {
  printf("Soy el proceso padre\n");
 }
 printf("Mensaje en ambos\n");
 exit(0);
}
```

4. Programe una aplicación que cree el árbol de procesos mostrado en el pizarrón. Para cada uno de los procesos creados (por copia exacta de código) se imprimirá en pantalla el pid de su padre, y cada proceso padre imprimirá los pids de sus hijos creados. Dibuje en papel el árbol creado usando los pids impresos por la aplicación desarrollada.
5. Programe una aplicación que cree seis procesos (por copia exacta de código). El primer proceso se encargará de realizar la suma de dos matrices de 10x10 elementos tipo entero, el segundo proceso realizará la resta sobre esas mismas matrices, el tercer proceso realizará la multiplicación de las matrices, el cuarto proceso obtendrá las transpuestas de cada matriz y el quinto proceso obtendrá las matrices inversas. Cada uno de estos procesos escribirá un archivo con los resultados de la operación que realizó. El sexto proceso leerán los archivos de resultados y los mostrará en pantalla cada uno de ellos.
   Programe la misma aplicación sin la creación de procesos, es decir de forma secuencial. Obtenga los tiempos de ejecución de cada una de las aplicaciones programadas, compare estos tiempos y dé sus observaciones.
6. Capture, compile y ejecute el siguiente programa de creación de un nuevo proceso con sustitución de un nuevo código, así como el programa que será el nuevo código a ejecutar. Observe su funcionamiento y experimente con el código.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
   pid_t pid;
   char *argv[3];
   argv[0]="/usuario/hola"; /*cambiar por su ruta */
   argv[1]="Desde el Hijo";
   argv[2]=NULL;
   if((pid=fork())==0)
   {
          printf("Soy el hijo ejecutando: %s\n", argv[0]);
          execv(argv[0],argv);
   }
   else
   {
          wait(0);
          printf("Soy el Padre\n");
          exit(0);
   }
}
```

```
/* hola.c Programa que será invocado */

#include<stdio.h>
int main(int argc, char *argv[])
{
 char mensaje[100];
 strcpy(mensaje,"Hola Mundo ");
 strcat(mensaje,argv[1]);
 printf("%s\n",mensaje);
 exit(0);
}
```

7. Programe una aplicación que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará tres procesos hijos más. Cada uno de los tres procesos generados ejecutará tres programas diferentes mediante sustitución de código, el primer programa evaluará una expresión aritmética (está aplicación es la que programó en la práctica 1), el segundo programa cambiará los permisos de un archivo dado (está aplicación es la que programó en la práctica 2), y el tercer programa obtendrá las matrices inversas que programó anteriormente. Observe el funcionamiento de su programa detalladamente y responda la siguiente pregunta ¿es posible un funcionamiento 100% concurrente de su aplicación? Explique porque sí o no es 100% concurrente su aplicación.
8. Programe la aplicación desarrollada en el punto 5 de la sección de Linux utilizando esta vez la creación de procesos por sustitución de código. Obtenga el tiempo de ejecución de la aplicación programada, compare este tiempo contra el tiempo obtenido del programa de creación de procesos por copia exacta de código, y dé sus observaciones.

Sección Windows:

1. Inicie sesión en Windows.
2. Investigue en MSDN la llamada al sistema CreateProcess(), WaitForSingleObject() y CloseHandle(), reporte su funcionalidad, argumentos y retorno.
3. Capture y compile el programa de creación de un nuevo proceso que a continuación se muestra.

```
#include <windows.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
   STARTUPINFO si;                    /* Estructura de información inicial para Windows */
   PROCESS_INFORMATION pi;      /* Estructura de información del adm. de procesos */
   int i;
   ZeroMemory(&si, sizeof(si));
   si.cb = sizeof(si);
   ZeroMemory(&pi, sizeof(pi));
   if(argc!=2)
   {
      printf("Usar: %s   Nombre_programa_hijo\n", argv[0]);
      return;
   }

   // Creación proceso hijo
   if(!CreateProcess(NULL, argv[1], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
   {
      printf( "Fallo al invocar CreateProcess (%d)\n", GetLastError() );
      return;
   }

   // Proceso padre
   printf("Soy el padre\n");
   WaitForSingleObject(pi.hProcess, INFINITE);

   // Terminación controlada del proceso e hilo asociado de ejecución
   CloseHandle(pi.hProcess);
   CloseHandle(pi.hThread);
}
```

4. Capture y compile el programa que contendrá al proceso hijo que a continuación se muestra.

```
#include <windows.h>
#include <stdio.h>

int main(void)
{
   printf("Soy el hijo\n");
   exit(0);
}
```

5. Ejecute el primer código pasando como argumento a través de la línea de comando, el nombre del archivo ejecutable del segundo código capturado. Observe el funcionamiento del programa, reporte sus observaciones y experimente con el código.
6. Compare y reporte tanto las diferencias como similitudes que encuentra con respecto a la creación de procesos por sustitución de código en Linux.
7. Programe una aplicación que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará 5 procesos hijos más. A su vez cada uno de los cinco procesos creará 3 procesos más. Cada uno de los procesos creados imprimirá en pantalla su identificador. **CONSEJO: INVESTIGUE LA FUNCIÓN GetCurrentProcessId() DEL API WIN32.**
8. Programe la aplicación desarrollada en el punto 5 de la sección de Linux utilizando esta vez la creación de procesos en Windows. Obtenga el tiempo de ejecución de la aplicación programada, compare este tiempo contra el tiempo obtenido del programa de creación de procesos por sustitución de código de Linux, y dé sus observaciones.