Procesos e Hilos

OCSO

Profesorado de Informática
CeRP del Suroeste, Uruguay

Bibliografía para los Temas 1 y 2

- Texto principal
  - Fundamentos de los sistemas operativos (Silberschatz, 2006, 7ª ed.)
    Capítulos 1 y 2
- A partir de esta clase van a necesitar salir a practicar bastante de C para familiarizarse. Esta clase apunta a ser una ayuda inicial.
  - Sistemas Operativos: aspectos internos y principios de diseño (Stallings, 2005, 5ª ed.)
    Capítulo 2
  - Fundamentos de Sistemas Operativos. Teoría y ejercicios resueltos
    (Candela, García, Quesada, Santana, Santos, 2007)
    Capítulo 1

Contenidos

- Qué es un proceso
- Estructuras de datos para gestionar procesos
- API para trabajar con procesos
- Hilos (threads). Descripción y gestión
- Algoritmos de planificación de la CPU

Bibliografía para el Tema 3

- Silberschatz, 7ª edición en español:
  – Capítulo 3 (procesos):
  3.1, 3.2, 3.3

  – Capítulo 4 (hebras/hilos):
  Entero, especial atención a 4.1 y 4.3

  – Capítulo 5 (planificación de la CPU):
  Entero, excepto 5.7

Ejemplos de llamadas al sistema

- Windows:
  – CreateProcess(): se indica el fichero ejecutable donde está el código del hijo

- UNIX:
  – fork(): crea un proceso hijo que es un duplicado del padre
  – exec(): sustituye el código por un nuevo fichero ejecutable (no crea un nuevo proceso)

Ejemplo de fork() y exec()

```
pid_t pid = fork();
if ( pid!=0 ){
      printf ("Soy el proceso padre");
      // otras acciones del  proceso     padre
}
else {
      // el hijo  entrará     por   aquí
      printf ("Soy el proceso hijo");
      //el hijo ejecuta la orden "ls -l”
      // que reemplaza el código actual del hijo
      execlp ("ls", "ls", "-l", NULL);
}
```

Terminación de procesos

- Un proceso termina cuando invoca a una llamada al sistema específica, ej. exit()
- También si se genera una excepción y el S.O. decide abortarlo.
- En UNIX, cuando un proceso termina, con él mueren sus descendientes.
- Podría existir una llamada al sistema para abortar otro proceso, ej. kill()
- Esperar por la terminación de un proceso hijo: wait()

Ejemplo de sincronización con wait()

```
pid_t pid = fork();
if ( pid!=0 ){
      printf ("Soy el padre, pid=%d\n", getpid());
      // otras acciones del proceso padre
      int status;
      pid_t hijo = wait (&status);
      // espera por el hijo
      // … el padre puede continuar
}
else {
      // el hijo  entrará     por   aquí
      printf ("Soy el hijo, pid=%d\n", getpid());
      // … otras acciones del hijo
      exit (1234);
}
```