Sistemas Operativos
Trabajo Práctico Nº 1

Creación de procesos y concurrencia

Objetivos

- Aplicar los conceptos de creación de procesos y concurrencia.
- Comprender el funcionamiento de las llamada al sistema fork/wait

Ejercicios

Crear un programa en C que, dado el nombre de un programa y una lista de argumentos, ejecute en paralelo una copia del programa por cada argumento.

El programa debe poder usarse desde línea de comandos y soportar la siguiente sintaxis:

```
parallel <nivel-concurrencia> <programa> <arg1> <arg2> … <argN>
```

y deberá ejecutar en paralelo los siguientes subprocesos:

```
programa <arg 1>
programa <arg 2>
programa <arg 3>
…
programa <arg n>
```

En ningún momento deberá superarse el nivel de concurrencia especificado como parámetro.

Ejemplo de uso:

Al ejecutar desde línea de comandos:

```
parallel 4 "ping -c 1" google.com yahoo.com bing.com gmail.com
twitter.com
```

El programa deberá ejecutar en paralelo los procesos "ping -c", creando como máximo 4 subprocesos al mismo tiempo:

```
ping -c 1 google.com
ping -c 1 yahoo.com
```

```
ping -c 1 bing.com
ping -c 1 gmail.com

...máximo nivel de concurrencia alcanzado, se debe esperar a
que
   termine algún proceso y luego...

ping -c 1 twitter.com
```

Otro ejemplo:

El siguiente comando:

```
parallel 8 gzip dir/*
```

deberá ejecutar en paralelo un proceso gzip por cada archivo contenido en el directorio “dir”. Respetando el máximo nivel de concurrencia 8.

Cuestionario

1) Utilizando el programa “parallel” que acaba de implementar, medir los tiempos de ejecución de un procesos de uso intensivo de cpu (por ejemplo gzip) y otro de uso intensivo de I/O (por ejemplo ping, ls, etc) variando los niveles de concurrencia de 1 a 8. Usar el comando ‘time’ de linux para realizar las mediciones.

Comparar los resultados en forma gráfica utilizando una planilla de cálculo. (Tiempo de ejecución total de los procesos en el eje Y, nivel de concurrencia en el eje X).

Nota: preferentemente realizar las pruebas en una computadora con más de un core.

Elaborar conclusiones.

2) Interceptar las llamadas al sistema operativo utilizando el comando ‘strace’. ¿ Qué llamadas al sistema se utilizan para la creación de los procesos ?

3) Ejecutar un programa lento (puede ser el comando sleep). Mientras el programa se encuentra en ejecución, examine los procesos. ¿ Cómo es la jerarquía de procesos ?

4) ¿ Qué ocurre si el padre no espera a sus hijos con ‘wait’ ? ¿ Qué ocurre con el “parent id” de los hijos cuando el proceso padre termina ?

Notas

Utilizar el compilador gcc con el estándar c99. Por ejemplo:

```
gcc -std=c99 -o parallel parallel.c
```