Universidad Católica Andrés Bello
Facultad de Ingeniería
Escuela de Informática

Materia: Sistemas de Operación
Profesor: Genesio De Stefano

Tarea 2: Creación de procesos, manejo de señales, tuberías

1) Elabore un programa en C que se encargue de crear procesos hasta colapsar el sistema. Cada proceso hijo debe mostrar la salida con datos útiles como su ID y hora de inicio (Puede agregar otras informaciones a la salida). Describa los resultados obtenidos, indique en el informe el tiempo transcurrido, número de procesos y cualquier otra información relacionada con la ejecución de su programa. La creatividad de la salida en pantalla formará parte de la evaluación.

2) Modifique el programa del ejercicio 'a' para que cada proceso hijo continúe creando a su vez nuevos procesos. Informe de los resultados usando los mismos parámetros del ejercicio 'a'.

3) Elabore un programa que cree un proceso hijo. El programa padre debe tener una variable inicializada en algún valor. Modifique el valor de esa variable en el proceso hijo y verifique si el valor ha cambiado en el proceso padre. Explique con detalle las conclusiones que obtuvo a partir de los resultados del ejercicio.

4) Señales. El bromista.

a) Elabore un programa que maneje las señales SIGHUP y SIGTERM. Cada vez que su programa reciba la señal SUGHUP debe crear un nuevo proceso que haga un exec llamando al programa xcalc. Su programa debe llevar cuenta de todos sus hijos creados. Cuando reciba la señal SIGTERM debe cerrar a todos sus hijos enviando a cada uno la señal SIGTERM y terminar su ejecución.

b) Ejecute su programa y envíele varias veces señales SIGHUP. Luego envíe una señal SIGKILL a su proceso. Explique con detalle las conclusiones que obtuvo a partir de los resultados del ejercicio.

5) Compile los siguientes programas

```
/* cpu-eater.c, compilar con -lm */
#include <stdio.h>
#include <math.h>
int main() {
        int i;
        double f;
        f=765476.0;
        for (i=1; i<3000000000; i++) {
                f = sqrt (f);
                f = f*f;
        }
}


/* io-eater.c */
#include <stdio.h>
```

```
#include <math.h>
int main() {
        FILE *fp;
        int i;
        double f;
        f=765476.0;
        for (i=1; i<30000000; i++) {
                fp=fopen("borra.txt", "w");
                fprintf(fp, "Testing...\n");
                fclose(fp);
        }
}
```

Ejecute 'io-eater' y con el comando 'strace -p' revise las llamadas al sistema de este programa. Ejecute 'cpu-eater' y con el comando 'strace -p' revise las llamadas al sistema de este programa.

a) Describa las llamadas al sistema que observó al ejecutar el ptrace.
b) Describa a nivel del sistema operativo (planificador de tareas) lo que ocurre con cada uno de estos procesos.

6) Compare el rendimiento de llamadas al sistema versus llamadas a procedimientos.

Escriba dos programas, El primero ejecutará un número elevado (cien mil o más) de llamadas a una procedimiento que sume un par de números. El segundo programa realizará una llamada al sistema getpid(), usando el mismo número de ciclos que en el programa anterior. Usando el comando time (man time) mida el tiempo de ejecución de cada programa.

a) Explique **en detalle** la diferencia de tiempos encontrada en ambos programas.

b) Explique las acciones que ocurren en el computador en ambos casos.

7) Tuberías (pipes) en UNIX
- Cree dos pipes (mkfifo) con nombres pipe1 y pipe2
- Ejecute los siguientes comandos:

```
$ echo -n x | cat - pipe1 > pipe2 &
$ cat <pipe2 > pipe1
```

a) Explique cada uno de los comandos (individuales) y los conectores usados. Explique qué hace cada una de esas líneas

Condiciones de entrega:

- Fecha de entrega: Viernes 18 de Octubre en clase.
- En grupos de 3 personas.
- Font: Arial 11
- Interlineado: 1.5
- Márgenes inferior, superior, derecha e izquierda: 2 cm
- Formato: En físico, sin carpeta y sin portada. Cada ejercicio debe **incluir el código fuente**.
- Incluya la bibliografía consultada según el formato APA.