PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Semestre 2019-1
Laboratorio 1

PARTE A

Si se ejecuta en la línea de comandos la siguiente orden:

```
ulimit -a
```

se obtiene la siguiente salida:

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 31367
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 2
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 31367
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Su objetivo será verificar algunos límites.

1) (2 puntos) Escriba un programa en lenguaje C, que obtenga descriptores de archivos hasta que el sistema se lo impida. El programa debió contar, en una variable, previamente los descriptores hasta obtener el error, en ese momento se debe imprimir la variable. Compare el valor con el límite mostrado arriba y escriba (como mensaje en el mismo archivo) sus comentarios.

2) (3 puntos) Escriba un programa en lenguaje C, que cree procesos hijos con la llamada al sistema fork(), hasta que el sistema se lo impida. El programa debió contar, en una variable, previamente los procesos hijos que pudo crear hasta obtener el error, en ese momento se debe imprimir la variable. Cada proceso hijo creado debe de hacer tiempo con la función de librería usleep(). La idea es que los procesos hijos se mantengan con vida hasta que el padre obtenga el error. Compare el valor con el límite mostrado arriba y escriba (como mensaje en el mismo archivo) sus comentarios.

PARTE B

Cuando se emplea la función de librería system, se pueden ejecutar comandos por el shell, pero las salidas de estos comandos ejecutados son enviadas a la pantalla. Si uno quisiera que la salida sea capturada por el mismo programa, tendríamos que usar al función de librería popen(). Junto con este archivo se adjunta un programa haciendo uso de popen().

3) (10 puntos) Elabore un programa en lenguaje C, que haga lo siguiente:

a) El proceso padre crea N hijos, formando un abanico de procesos. Se comunica con cada uno de ellos a través de un *pipe* (es decir hay N *pipes*). El valor de N es definido como una macro con valor 4.

b) Cada proceso hijo, una vez creado, lee del *pipe*. Al inicio se quedará bloqueado.

c) El proceso padre ejecuta "*ls*"+ *path* con la función de librería `popen()`. Donde *path* es el valor de una ruta absoluta ingresada en la línea de comandos (emplee `char *argv[]`).

d) El proceso padre lee los nombre de los archivos de *path*, los procesa y los deposita uno a uno en los *pipes*, de forma rotativa hasta acabar con la lista.

e) Cada proceso hijo toma el nombre (con su ruta absoluta) del *pipe* y abre el archivo (use *open*) y lo lee (use *read*) luego crea otro archivo localmente en directorio actual y graba todo su contenido en este nuevo archivo (use *write*).

Lo que se debe obtener es la copia de todos los archivos de *path* al directorio actual.

Usted debe resolver las situaciones para que el programa termine dejar procesos bloqueados o procesos zombis.

4) (5 puntos) Elabore un programa en lenguaje C, que implemente su propio popen con la opción "r". Debe funcionar de forma análoga al programa ejemplo proporcionado.

Porf. Alejandro T. Bello Ruiz