

MiniTP de Shell y Procesos

Sistemas Operativos y Redes

Ejercicio 1: Shell y terminal

En este punto se pide realizar un script de shell que reciba una palabra como parámetro, cree una carpeta en el directorio home con una lista de todos los archivos que terminan con extensión .txt de la computadora, junto a sus permisos y la fecha actual dentro de dicha carpeta. Al finalizar el script se muestra el contenido de la lista por consola

La principal dificultad en este ejercicio fue el uso de los comparadores en los if ya que no encontraba uno se que adecue a mi implementación pude resolverlo implementando `if [-z "$folder"]`; el operador -z verifica si una cadena está vacía, si lo está devuelve true.

Otra dificultad fue el poder resolver como leer los archivos terminados en extensión .txt de la computadora `ls -lR / | grep '\.txt$' > "text.txt"` se lista todos los archivos y directorios del sistema de archivos a partir del directorio raíz (/). Y se utiliza un | (pipe) para redirigir la salida del comando hacia grep '\.txt\$', que filtra por las líneas que contienen nombres de archivos que terminan con la extensión .txt. y haciendo uso de > se escriben en text.txt.

Ejercicio 2: Estados de un Proceso

En este punto se pide realizar un programa en C compuesto de instrucciones que realizan operaciones aritméticas y operaciones de I/O, para visualizar con htop como el programa cambia de estados.

La principal dificultad de este ejercicio fue el uso de NINA con htop, ya que no logré poder ejecutar htop y ver los cambios. Para resolverlo tuve que utilizar VirtualBox con Ubuntu y correr dos terminales a la vez, ejecutando htop mientras el programa esperaba la segunda entrada por teclado.

```

fran@fran-VirtualBox: ~/Desktop
fran@fran-VirtualBox:~/Desktop$ ./punto2
ingrese un numero: 40
ingrese otro numero: 30
El resultado es: 1200
fran@fran-VirtualBox:~/Desktop$ ./punto2
ingrese un numero: 40
ingrese otro numero: 

```

```

fran@fran-VirtualBox: ~/Desktop
Tasks: 124, 293 thr: 1 running
Load average: 0.28 0.39 0.31
Uptime: 06:28:45
Mem[|||||] 940M/3.83G
Swp[|||||] 14.9M/2.62G

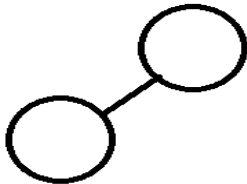
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2441	fran	20	0	703M	19032	8600	S	0.0	0.5	0:00.00	/usr/bin/snap userd
4646	root	20	0	243M	7188	6408	S	0.0	0.2	0:00.00	/usr/libexec/boltd
4648	root	20	0	243M	7188	6408	S	0.0	0.2	0:00.00	/usr/libexec/boltd
30053	fran	20	0	2818M	73212	53660	S	0.0	1.8	0:00.00	gjs /usr/share/gnome-shell/extensions/ding@rastersoft.com
30067	fran	20	0	2818M	73212	53660	S	0.0	1.8	0:00.00	gjs /usr/share/gnome-shell/extensions/ding@rastersoft.com
30522	fran	20	0	790M	74700	52084	S	0.0	1.9	0:00.00	/usr/bin/nautilus --gapplication-service
30525	fran	20	0	790M	74700	52084	S	0.0	1.9	0:00.00	/usr/bin/nautilus --gapplication-service
30527	fran	20	0	790M	74700	52084	S	0.0	1.9	0:00.00	/usr/bin/nautilus --gapplication-service
30683	root	20	0	168M	11324	9904	S	0.0	0.3	0:00.00	/usr/sbin/cups-browsed
30686	root	20	0	168M	11324	9904	S	0.0	0.3	0:00.00	/usr/sbin/cups-browsed
30702	root	20	0	391M	30756	25988	S	0.0	0.8	0:00.00	/usr/libexec/fwupd/fwupd
31299	fran	20	0	553M	56268	42848	S	0.0	1.4	0:00.00	/usr/libexec/gnome-terminal-server
31301	fran	20	0	553M	56268	42848	S	0.0	1.4	0:00.00	/usr/libexec/gnome-terminal-server
31302	fran	20	0	553M	56268	42848	S	0.0	1.4	0:00.00	/usr/libexec/gnome-terminal-server
31379	fran	20	0	2772	960	868	S	0.0	0.0	0:00.00	./punto2

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort By F7 Nice F8 Nice F9 Kill F10 Quit

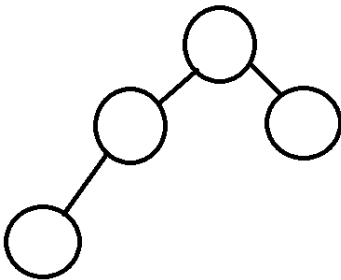
Ejercicio 3: Procesos y Fork

El programa crea un proceso principal (padre) y luego entra en un bucle que se ejecuta tres veces por $n=3$. En cada iteración, se llama a `fork()`, lo que da como resultado la creación de un nuevo proceso hijo. La llamada a `fork()` duplica el proceso en dos procesos independientes y cada uno de ellos continúa la ejecución desde el punto en que se llamó a `fork()`.



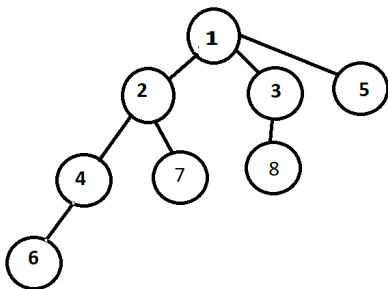
Iteración 1:

El proceso padre crea un nuevo proceso hijo mediante `fork()` y ambos procesos imprimen "Soy un proceso!".



Iteración 2:

Cada uno de los procesos crea otro proceso hijo y los cuatro procesos imprimen "Soy un proceso!".



Iteración 3:

Cada uno de los cuatro procesos crea otro proceso hijo. Los ocho procesos imprimen "Soy un proceso!".

```

sorcluser23@Nina: ~/sor/miniTPShellProcesos
|      int
p3.c:7:10: error: 'i' undeclared (first use in this function)
  7 |     for (i=0; i<n; i++){
    |         ^
p3.c:7:10: note: each undeclared identifier is reported only once for each function it appears in
sorcluser23@Nina:~/sor/miniTPShellProcesos $ nano p3.c
sorcluser23@Nina:~/sor/miniTPShellProcesos $ gcc -o p3 p3.c
sorcluser23@Nina:~/sor/miniTPShellProcesos $ ./p3
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
Soy un proceso!
sorcluser23@Nina:~/sor/miniTPShellProcesos $

```

Ejercicio 4: Threads

En este punto se pide realizar la comparación de tiempo en la ejecución de una función 5 veces, ejecutándola una atrás de otra en un programa y mediante hilos en otro. Se observa una gran diferencia en el tiempo de ejecución de ambos programas. Ya que, cuando se utilizan hilos en un programa, estos pueden ejecutarse en paralelo o en concurrencia, dependiendo del hardware. Esto significa que varios hilos pueden ejecutar sus rutinas al mismo tiempo o de manera intercalada, reduciendo el tiempo de ejecución significativamente.

```

sorcluser23@Nina: ~/sor/miniTPShellProcesos
sorcluser23@Nina:~/sor/miniTPShellProcesos $ time ./p4

real    0m10,009s
user    0m0,005s
sys     0m0,001s
sorcluser23@Nina:~/sor/miniTPShellProcesos $ time ./p4thread

real    0m2,007s
user    0m0,007s
sys     0m0,002s
sorcluser23@Nina:~/sor/miniTPShellProcesos $

```