

MiniTP de Shell y Procesos

Sistemas Operativos y Redes

Condiciones para Aprobar

Para la evaluación del presente trabajo, deben realizar los siguientes puntos:

- El trabajo es individual
- Enviar el trabajo en formato digital dentro de los plazos establecidos. Adjuntar el código fuente de su implementación y un informe del trabajo realizado punto por punto, dificultades encontradas y soluciones propuestas.

Puntaje / Calificación:

El presente trabajo se califica con las notas:

- ★ I (insuficiente)
- ★ A- (aprobado menos, no puede tener dos A- en la cursada),
- ★ A (aprobado)
- ★ A+ (aprobado más, redondea para arriba la nota final en caso de promocionar)

Recuperatorio

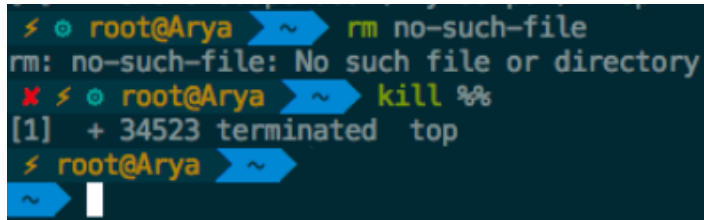
En caso de no aprobar tiene un plazo de una semana para entregar el TP con las correcciones indicadas más ejercicios adicionales que se agregaran al enunciado. En recuperatorio no se pone A +.

Introducción

Los objetivos de este trabajo son:

- Familiarizar al alumno con la línea de comandos.
- Familiarizar al alumno con las nociones de Proceso y Thread.

Ejercicio 1 : Shell y terminal



```
> root@Arya ~$ rm no-such-file
rm: no-such-file: No such file or directory
> root@Arya ~$ kill %1
[1] + 34523 terminated top
> root@Arya ~$
```

- Realice un script de shell tal que realice las siguientes tareas:
 - Debe recibir por parámetro una palabra y debe crear un directorio con dicho nombre en tu home de usuario.
 - Dentro de ese directorio debe crear un archivo .txt
 - Debe agregar al archivo anterior el listado de todos los archivos de la computadora que terminan con la extensión .txt, y además de los nombres de los archivos se tienen que ver los permisos de los mismos.
 - Al final del archivo y del listado debe agregar la fecha y la hora del sistema
 - Cuando el script termine debe mostrar por pantalla el contenido del archivo.
- Puede usar los siguientes operadores y comandos:
 - pipe | ,
 - redirección > ,
 - redirección concatenando >>
 - y los comandos
 - grep
 - ls
 - cat

Ejercicio 2: Estados de un Proceso

En esta parte vamos a aplicar nuestros conocimientos de procesos y sus estados.

- Realizar un programa en C compuesto de instrucciones que realizan cálculos (operaciones aritméticas) y operaciones de I/O (leer un input del usuario). Compilar y ejecutar su programa y visualizar los estados por los que pasa. Puede usar la herramienta **htop**.
- Ejecutar su programa y demostrar mediante capturas de pantalla del programa **htop** que su programa efectivamente cambia de estados.

```

 1 [ |          1.3%]   Tasks: 110, 548 thr; 1 running
 2 [ |          0.7%]   Load average: 0.55 0.48 0.31
 3 [ ||         2.6%]   Uptime: 04:10:18
 4 [ ||         2.0%]
 Mem[|||||2.55G/3.66G]
 Swp[||        178M/3.79G]

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	135M	4048	2640	S	0.0	0.1	0:01.53	/sbin/init
2326	andrew	20	0	9620M	548M	99M	S	0.0	14.6	20:30.05	/usr/lib/firef
4571	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.01	/usr/lib/f
3895	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.32	/usr/lib/f
3734	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.47	/usr/lib/f
3726	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.48	/usr/lib/f
3446	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:01.26	/usr/lib/f
3445	andrew	20	0	2249M	539M	88508	S	0.0	14.4	9:22.14	/usr/lib/f
4536	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.02	/usr/lib
3502	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.06	/usr/li
3501	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.66	/usr/li
3500	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.67	/usr/li
3499	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.63	/usr/li
3496	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.01	/usr/li

F1Help F2Setup F3Search F4Filter F5Sorted F6Collap F7Nice -F8Nice +F9Kill F10Quit

Ejercicio 3 : Procesos y Fork

- Compilar, ejecutar el siguiente programa. Describir y justificar el output. Dibujar el árbol de proceso padre-hijo que se van generando para $n=3$

Ejercicio 4 : Threads

El siguiente programa ejecuta la función `do_nothing()` cinco veces. Esta función sólo espera 2 segundos y continúa:

```
#include <stdio.h> //incluimos la libreria de estandar input/output
#include <unistd.h> //para hacer sleep
#include <time.h>   //para inicializar el tiempo

void do_nothing(int microseconds){
    usleep(2000000); //esperar 2 segundos, 1 millon de microsegundos en 1 segundo
                    //dormir el proceso, simula que esta haciendo alguna tarea
}

int main() {
    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();
    return 0;
}

//para compilar: gcc do_nothing.c -o ejecutable
//para ejecutar: ./ejecutable
```

- Con la función `time`, medir el tiempo que tarda el programa anterior.
- Modificar el programa anterior para que cada una de las 5 llamadas a la función `do_nothing()` se ejecute por un thread.
- Medir el tiempo que tarda su nuevo programa. Qué diferencias observa? Porque?

Fin del MiniTP