

Mini TP - Shell y Procesos

Propósito y sentido de la actividad	1
Producto final de la actividad	1
Evaluación y entrega	2
Software necesario para el TP	2
Ayuda y consultas	2
Enunciado del Mini TP	2
Ejercicio 1 : Shell y terminal	2
Ejercicio 2 : Estados de un Proceso	3
Ejercicio 3 : Threads	4
Anexo	4

Propósito y sentido de la actividad

En este tp se practican los comandos más usados del sistema GNU/Linux. Conocer y ganar práctica en estos comandos será útil a lo largo de la materia así como también en la vida profesional donde el uso de este sistema puede ser excluyente e indispensable.

En este tp también se trabaja con el concepto de proceso y sus estados de un proceso. Estos conceptos son la base para entender el funcionamiento de un sistema operativo y eventualmente mejorar su desempeño y performance.

Producto final de la actividad

Al finalizar esta actividad tendremos:

- Un ejemplo de shell script Bash que nos permitirá avanzar sobre scripts más complejos. Podremos darle permisos y ejecutarlo.
- Tendremos ejemplos básicos de programas escritos en el lenguaje C, sabremos compilarlos y ejecutarlos dentro del sistema GNU/Linux.

Evaluación y entrega

Esta actividad es individual, obligatoria y con autoevaluación. La fecha de entrega se encuentra en el calendario, ese mismo día se publicará la solución para que pueda realizar la autoevaluación.

Fecha de entrega: Ver calendario

Espacio de entrega: Por Moodle

Software necesario para el TP

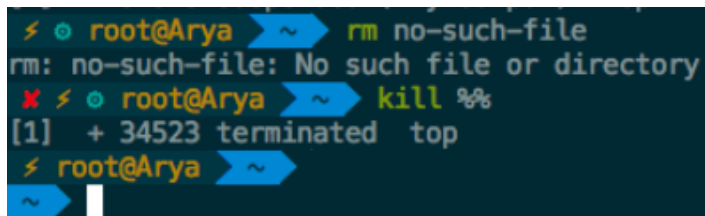
Para realizar esta actividad es necesario tener instalado un sistema GNU/Linux sobre una computadora con procesador de al menos dos núcleos.

Ayuda y consultas

Para realizar consultas sobre algún error durante los experimentos se solicita enviar una captura de pantalla del error y la secuencia de pasos que realizaron para obtener ese error. Tendremos disponible un canal de consultas en Telegram, así como también el foro de consultas en moodle y los mails de los profesores.

Enunciado del Mini TP

Ejercicio 1 : Shell y terminal



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

- Realice un script de shell llamado **miniTP.sh** tal que realice las siguientes tareas:
 - Al momento de ejecutarse, el programa debe recibir por parámetro en la línea de comandos un nombre y debe crear un directorio con dicho nombre en el home del usuario.

Por ejemplo, la forma de ejecutarlo debe ser así: `./miniTP.sh pepito`

Para acceder al home puede ejecutar: `cd $HOME`

Si se ejecuta sin parámetros debe dar un mensaje de error, por ejemplo:

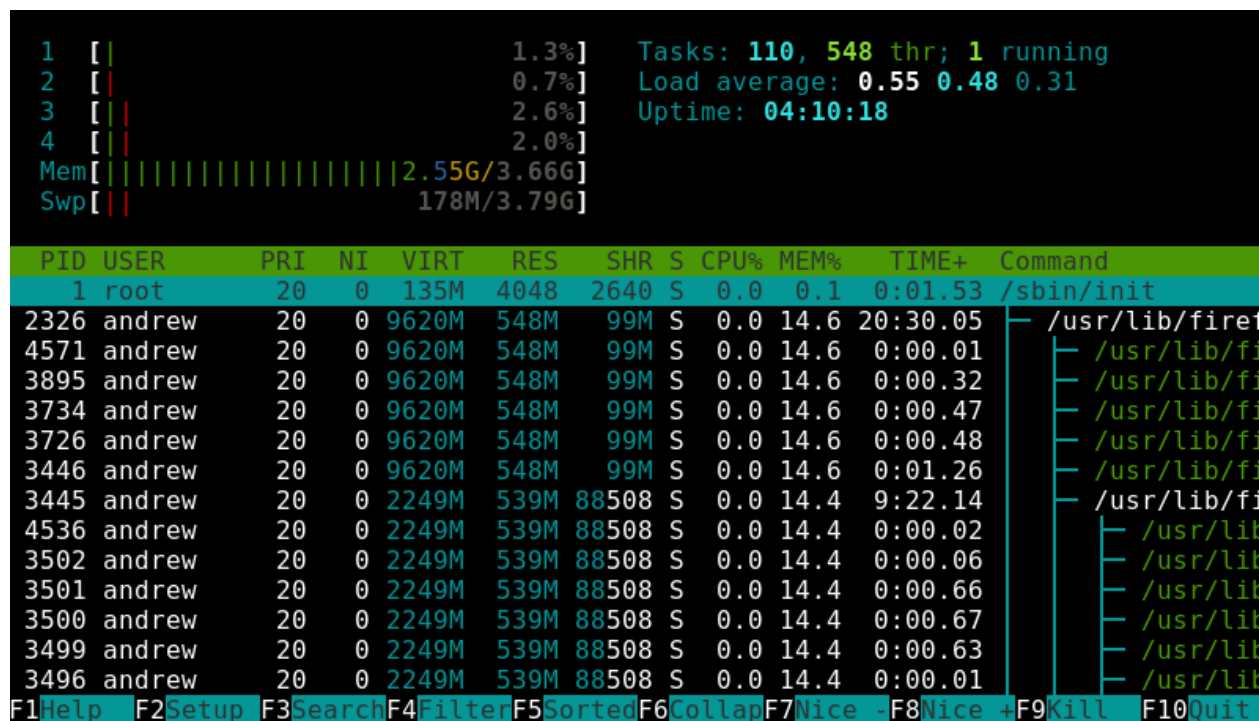
“Error: debe ejecutar con un parámetro al menos”

- Dentro del directorio creado en el punto anterior debe crear un archivo llamado **contenido_home.txt**
- Debe agregar al archivo anterior el listado de todos los archivos del home (incluidos los archivos ocultos si existe alguno) y además de los nombres de los archivos se tienen que ver los permisos de los mismos.
- Al final, el script **miniTP.sh** debe mostrar por pantalla el contenido del archivo **contenido_home.txt**, esperar a que el usuario aprete enter y después terminar.

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 comprobar mediante el programa htop que su programa efectivamente cambia de estados.



The screenshot shows the htop interface. At the top, system statistics are displayed: 1 task running (1.3%), 548 threads, 1 running task, load average of 0.55, 0.48, and 0.31, uptime of 04:10:18, memory usage of 2.55G/3.66G, and swap usage of 178M/3.79G. Below this is a table of running processes. The first process is root (PID 1) in state S, using 135M virtual memory and 4048K resident memory. Other processes include andrew (PID 2326) and several instances of andrew (PIDs 4571, 3895, 3734, 3726, 3446, 3445, 4536, 3502, 3501, 3500, 3499, 3496) all in state S, using 9620M virtual memory and 548M resident memory. The bottom of the screen shows function keys: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Sorted, F6 Collap, F7 Nice -, F8 Nice +, F9 Kill, and F10 Quit.

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/fire
4571	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.01	/usr/lib/fi
3895	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.32	/usr/lib/fi
3734	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.47	/usr/lib/fi
3726	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:00.48	/usr/lib/fi
3446	andrew	20	0	9620M	548M	99M	S	0.0	14.6	0:01.26	/usr/lib/fi
3445	andrew	20	0	2249M	539M	88508	S	0.0	14.4	9:22.14	/usr/lib/fi
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/lib
3501	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.66	/usr/lib
3500	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.67	/usr/lib
3499	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.63	/usr/lib
3496	andrew	20	0	2249M	539M	88508	S	0.0	14.4	0:00.01	/usr/lib

Ejemplo de una captura de htop

Ejercicio 3 : Threads

El siguiente programa ejecuta la función `calculo_aritmetico()` cinco veces. Esta función realiza un operación aritmética que requiere cierto tiempo 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 calculo_aritmetico(){
    int contador=0;
    while(contador < 2147483647){
        contador=contador+1;
    }
}

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

//para compilar: gcc calculo_aritmetico.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 `calculo_aritmetico()` como un único hilo de ejecución.
- Medir el tiempo que tarda su nuevo programa. ¿Qué diferencias observa en el tiempo?
¿Por qué es importante la cantidad de núcleos en el procesador?

Fin del MiniTP

Anexo

Consultas frecuentes del mintp: [link](#)