Trabajo Práctico N1

rdproc

Objetivos 1

Introducción 1

Step: A rdproc, versión inicial 1

Step B: Command line arguments 2

Step C: 2

Step D: Process file descriptor 3

Ejemplo de ejecución 3

Testing 3

Preguntas 3

Criterios de Corrección 4

Qué se debe Entregar 4

Objetivos

Con el desarrollo del siguiente Trabajo Práctico (TP), se busca:

- Realizar un primer programa en C para GNU/Linux.
- Aprender sobre /proc: qué es, qué información contiene, cómo se puede utilizar.
- Aprender a compilar con GNU gcc, debuggear con GNU gdb.
- Dirigir el proceso de compilación a través de GNU Make.

Introducción

Este trabajo práctico consta en la elaboración de un programa en lenguaje C sobre GNU/Linux. El trabajo se divide en soluciones incrementales.

Step: A rdproc, versión inicial

Crear la siguiente aplicación, llamada **rdproc**

Escriba una versión inicial del programa que inspeccione las variables del kernel a través de /proc e informe por stdout:

- Tipo y modelo de CPU.

- Versión de Kernel.
- Cantidad de tiempo transcurrido desde que se inició el sistema operativo, con el formato ddD hh:mm:ss.
- Cantidad de sistemas de archivo soportados por el kernel

También se pide incluir una cabecera donde se indique el nombre de la máquina y la fecha y
hora actuales.

Step B: Command line arguments

Escriba una segunda versión del programa que imprima la misma información que la versión por defecto, pero en caso de invocarse con la **opción -s**, agrega la siguiente información:

- Cantidad de tiempo de CPU utilizado para usuarios, sistema y proceso idle.
- Cantidad de cambios de contexto.
- Número de procesos creados desde el inicio del sistema.

Step C:

La tercera parte involucra imprimir todo lo de las versiones anteriores, pero cuando se invoca con la **opción -l** interval duration imprime además:

- Número de peticiones a disco realizadas.
- Cantidad de memoria configurada en el hardware.
- Cantidad de memoria disponible.
- Lista de los promedios de carga de 1 minuto.

Asi por ejemplo rdproc -l 2 100 mostrará el promedio de carga de 1 minuto por 100 segundos tomando muestras en intervalos de 2 segundos. El comando rdproc -l a b, lee los datos indicados arriba (peticiones de disco, cantidades de memoria, promedios de carga) de /proc, y los imprime repetidamente cada a segundos; esto se repite hasta que hayan pasado b segundos.

Por ejemplo:
$ rdproc -l 5 15
Peticiones a disco: 12345
Memoria disponible / total: 1000000 / 8000000
Promedio de carga en el último minuto: 0.88
[Pausa de 5 segundos]

Peticiones a disco: 12348
Memoria disponible / total: 2000000 / 8000000
Promedio de carga en el último minuto: 0.82
[Pausa de 5 segundos]
Peticiones a disco: 12645

Memoria disponible / total: 500000 / 8000000
Promedio de carga en el último minuto: 0.98
[Pausa de 5 segundos]
Peticiones a disco: 99995
Memoria disponible / total: 300000 / 8000000
Promedio de carga en el último minuto: 1.07
[so2004@hal so2004]$
En particular noten que el promedio de carga de un minuto es un valor que el sistema provee ya calculado, no es algo que ustedes tengan que calcular.
Notar que cada opción incluye a la otra, por lo que rdproc -s -l 2 100 no debería ser aceptado y en tales casos resulta útil imprimir por la salida estándar un resumen de las opciones aceptadas.

Step D: Process

Agregar a **rdproc**, la siguiente opción -p <pid>. Con esta nueva opción, además, se imprime información sobre los file descriptors a los que tiene acceso un proceso en particular.
Agregar la opción -f <pid>, para imprimir los límites de archivos abiertos para un cierto proceso.
Finalmente, agregar la opción -t para imprimir el kernel stack trace actual de un cierto proceso. Imprimir solamente el nombre del símbolo.

Ejemplo de ejecución

% rdproc -p 123
<rdproc previous info>
<permits> <file type> <path to fd>
...
<permits> <file type> <path to fd>

Testing

1) Testear con un programa que abra un archivo y se quede en un loop infinito.
2) Utilizando shell:
   Primero, se puede conseguir el process ID usando el comando ps:
   % ps aux | grep processName

   Opcionalmente, se puede instalar y utilizar el command pidof:
   % pidof processName

   Luego, podemos acceder a /proc/<id>/fd

   Finalmente, un comando que resume esta consigna:

% lsof -a -p <pid>

3) Testear la opción -f con el comando ‘ulimit’.

Preguntas

1) Cuáles son los tipos de type descriptors que podemos encontrar en /proc/<id>/fd?
2) Suponiendo que un usuario está ejecutando el proceso pid 1212, es válida la ejecución del siguiente comando desde una terminal nueva?
   % echo “Hello, world.” >> /proc/1212/fd/1
3) Qué diferencia hay entre hard y soft limits?

Criterios de Corrección

- Se debe compilar el código con los flags de compilación: -Wall -Pedantic
- Dividir el código en módulos de manera juiciosa.
- Estilo de código.
- El código no debe contener errores, ni warnings.
- El código no debe contener errores de cppcheck.

Qué se debe Entregar

- Informe del desarrollo del proyecto.
- Código (funcionando bajo las especificaciones dadas y bajo cualquier caso de test de parámetros de entrada).
- Makefile