Sistemas Operativos

Segundo Cuatrimestre de 2008

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo práctico

Grupo 8

Abstract

Investigación e implementación de distintas funcionalidades de un Ubuntu JeOS corriendo sobre una máquina virtual (Virtual BOX). Threads, pipes, fork, join, acceso al kernel mediante módulos. Manejo de la consola y organización de archivos y carpetas del Sistema Operativo linux.

Palabras clave

Ubuntu JeOS, Módulo del kernel, Threads

Integrante	LU	Correo electrónico
Matías López y Rosenfeld	437/03	matiaslopez@gmail.com
Román Gorojovsky	530/02	rgorojovsky@gmail.com
Facundo Ciccioli	399/03	facundofc@gmail.com

Índice

1. Introducción teórica

2. Desarrollo

2.1. Implementación

3. Resultados

4. Discusión

5. Conclusiones

6. Algoritmos y código relevante

Se presentan a continuación las partes del código más importantes: Generación de un Trazador Cúbico a partir de una tabla de puntos y la función que dice si un auto derrapa en una trayectoria.

6.1. Splines Paramétricos

```
1 ParOrdenado Param::evaluar(double t) const
2 {
3     /* Buscar en los splines x e y la evaluacion en t */
4     return ParOrdenado(_x.evaluar(t),_y.evaluar(t));
5 }
```

7. Consignas

Antes de empezar, ejecute:

```
sudo apt-get install man-db manpages manpages-dev
```

De esta manera tendrá acceso a ayuda en línea ejecutando:

```
man <comando>
```

Por ejemplo:

```
man cp
```

Puede además instalar la versión en castellano de la ayuda ejecutando:

```
sudo apt-get install manpages-es
```

Para acceder a la ayuda en castellano ejecute por ejemplo:

```
man -L es cp
```

Tenga presente que no todos los comandos poseen ayuda en castellano.

`sudo` permite a usuarios normales ejecutar comandos que requieren permisos de administrador. Al ejecutar un comando con `sudo` el sistema le pedirá su password, y no el password del administrador (llamado `root` en Linux, siguiendo la tradición de Unix). Esto sucede ya que el sistema permite que ciertos usuarios (que deberían corresponderse con usuarios "privilegiados" del sistema) puedan utilizar `sudo` ingresando solamente su propio password. El usuario por defecto creado en una instalación de Ubuntu tiene este permiso y por lo tanto en Ubuntu no es necesario una cuenta de administrador o `root`.

`apt-get` es el manejador de paquetes de la distribución Ubuntu. Permite instalar, actualizar y desinstalar programas. Más adelante lo utilizaremos para instalar las herramientas necesarias para compilar programas en Linux.

Si se encontrara detrás de un proxy, antes de utilizar `apt-get` debe configurar el proxy. Ejecute el siguiente comando:

```
sudo echo "Acquire::http::Proxy \"http://proxy.uba.ar:8080\";" > /etc/apt/apt.conf
```

7.1. Comandos básicos de Unix

1. `pwd` Indique qué directorio pasa a ser su directorio actual si ejecuta:
   a) `cd /usr/bin`
   b) `cd`
   c) ¿Cómo explica el punto anterior?
2. `cat` ¿Cuál es el contenido del archivo `/home/<usuario>/.profile`?
3. `find` Liste **todos** los archivos que comienzan con `vmlinuz`.
   Estos archivos son imágenes del kernel Linux.
4. `mkdir` Genere un directorio `/home/<usuario>/tp`.
5. `cp` Copie el archivo `/etc/passwd` al directorio `/home/<usuario>/tp`.
6. `chgrp` Cambie el grupo del archivo `/home/<usuario>/tp/passwd` para que sea el suyo.
7. `chown` Cambie el dueño del archivo `/home/<usuario>/tp/passwd` para que sea su usuario.
8. `chmod` Cambie los permisos del archivo `/home/<usuario>/tp/passwd` para que:
   - el propietario tenga permisos de lectura, escritura y ejecución
   - el grupo tenga sólo permisos de lectura y ejecución
   - el resto tenga sólo permisos de ejecución

9. `grep`

   Muestre las líneas que tienen el texto "localhost" en el archivo `/etc/hosts`.

   Muestre todas las líneas que tengan el texto "POSIX" de todos los archivos (incluyendo subdirectorios) en `/etc`. Evite los archivos binarios y aquellos archivos y directorios que no tienen permiso de lectura para su usuario.

10. `passwd` Cambie su password.

11. `rm` Borre el archivo `/home/<usuario>/tp/passwd`

12. `ln`

    Enlazar el archivo `/etc/passwd` a los archivos `/tmp/contra1` y `/tmp/contra2`.

    Hacer un `ls -l` para ver cuantos enlaces tiene `/etc/passwd`.

    Estos enlaces se llaman "hardlinks". Cada nuevo enlace referencia el mismo espacio ocupado del disco rígido, y por lo tanto cada hardlink es igual de representativo de esos bytes ocupados del disco rígido. El espacio ocupado solamente se liberará cuando todos los enlaces hayan sido borrados.

    Ahora enlace el archivo `/etc/passwd` de manera "soft" al archivo `contra3`.

    Verifique con `ls -l` que no aumentó la cantidad de enlaces de `/etc/passwd`.

    Estos enlaces se llaman "softlinks" y apuntan no a los bytes del disco rígido sino a la ruta del archivo a ser enlazado. Operar sobre el softlink es igual que operar sobre el archivo, sin embargo los softlinks no cuentan en la cantidad de enlaces (ya que no apuntan a los bytes ocupados del disco rígido) y pueden ser borrados sin afectar al archivo original, aunque si se borra el archivo original el softlink quedará huérfano y no apuntará a nada.

13. `mount`

    Monte el CD-ROM de instalación de Ubuntu JeOS y liste su contenido.

    Para hacer esto deberá especificar la ISO de instalación de Ubuntu JeOS como CD-ROM de la máquina virtual. Si bien puede hacer esto como lo hizo para instalar el sistema, si la máquina virtual está corriendo debe hacer click derecho en el ícono con forma de CD-ROM en la esquina inferior derecha de la máquina virtual, y seleccionar **CD/DVD-ROM Image...** (ver figura **??**). En la ventana que aparece seleccione la ISO de instalación (ver figura **??**).

    Presente los filesystems que tiene montados.

14. `df` ¿Qué espacio libre tiene cada uno de los filesystems montados?

15. `ps` ¿Cuántos procesos de usuario tiene ejecutando? Indique cuántos son del sistema.

16. `umount` Desmonte el CD-ROM de instalación de Ubuntu JeOS.

17. `uptime` ¿Cuanto tiempo lleva ejecutando su máquina virtual?

18. `uname` ¿Qué versión del kernel de Linux está utilizando?

7.2. Salida estándar y pipes

1. STDOUT

   a) Conserve en el archivo `/home/<usuario>/tp/config` la salida del comando `ls` que muestra todos los archivos del directorio `/etc` y de los subdirectorios bajo `/etc`.

   b) Presente cuantas líneas, palabras y caracteres tiene `/home/<usuario>/tp/config`.

   c) Agregue el contenido, ordenado alfabéticamente, del archivo `/etc/passwd` al final del archivo `/home/<usuario>/tp/config`.

   d) Presente cuantas líneas, palabras y caracteres tiene `/home/<usuario>/tp/config`.

2. Pipes

   a) Liste en forma amplia los archivos del directorio `/usr/bin` que comiencen con la letra "a". Del resultado obtenido, seleccione las líneas que contienen el texto `apt` e informe la cantidad de caracteres, palabras y líneas.

   Está prohibido, en este ítem, usar archivos temporales de trabajo.

7.3. Scripting

Escriba un script de shell que sincronice dos carpetas. El script debe recibir las dos carpetas (origen y destino, en ese orden) desde la línea de comando, o preguntarlas interactivamente al usuario en caso de no recibirlas. Además, debe aceptar un modificador `-r` que indica modo de operación recursivo.

Una vez conocidas las dos carpetas con las que se operará, el script deberá copiar todos los archivos de la carpeta origen que no estén presentes en la carpeta destino, y además también deberá copiar todos los archivos presentes en ambas carpetas que tengan una fecha de modificación posterior en la carpeta origen que en la carpeta destino. De esta manera, al ejecutar el script, estará seguro de que la carpeta destino contiene toda la información de la carpeta origen, excepto lo que fue modificado posteriormente en la carpeta destino.

El modificador `-r` indica al script realizar la sincronización también con los archivos de todos los subdirectorios de la carpetas origen y destino.

Sugerencia Recuerde que generalmente el espíritu de los scripts es proveer la mínima lógica necesaria alrededor de otros programas ya presentes en el sistema que puedan proveer parte de la funcionalidad requerida para resolver un determinado problema.

7.4. Ejecución de procesos en background

Antes de resolver esta sección instale los siguientes paquetes en la máquina virtual:

- `nano`: editor de texto.
- `mc`: manejador de archivos.
- `gcc`: compilador de C.
- `libc6-dev`: biblioteca estándar de C.

Cree el archivo `/home/<usuario>/tp/loop.c`. Compílelo con `gcc`. El programa compilado debe llamarse `loop`.

7.4.1. loop.c

1. Correrlo en foreground. ¿Qué sucede? Mate el proceso con `Ctrl-c`.
2. Ahora ejecútelo en background: `/usr/src/loop > /dev/null &`. Mate el proceso con el comando `kill`.

7.5. IPC y sincronización

7.5.1. Pipes

Muestre con un ejemplo en lenguaje C como realizar exclusión mutua entre dos procesos utilizando un sólo pipe.

Sugerencia Revise la ayuda de la llamada al sistema `pipe` para construir el pipe y de `fork` para crear nuevos procesos.

7.5.2. Threads

Antes de resolver este ejercicio instale el paquete `glibc-doc`.

Resuelva el problema de productor/consumidor utilizando threads.

Sugerencia Revise la ayuda de `pthreads`, la implementación de threads en Linux, para conocer los mecanismos de creación y destrucción de threads. Además, `pthreads` provee mecanismos de sincronización que le ayudarán a resolver este ejercicio.

7.6. El kernel Linux

Antes de resolver esta sección instale los siguientes paquetes en la máquina virtual:

- `make`: utilidad para mantener grupos de programas.
- `linux-headers-<version>`: headers del kernel de Linux.

Sustituya `<version>` por el resultado del comando `uname -r`.

7.6.1. Funcionamiento del kernel Linux

1. Describa la administración del procesador utilizada por defecto en el kernel Linux.
2. Describa la administración de memoria utilizada por defecto en el kernel Linux.
3. Describa el sistema de archivos utilizado en la distribución de Linux que instaló en la máquina virtual. ¿Qué capas existen en el kernel Linux para soportar sistemas de archivos sobre dispositivos de bloques?

7.6.2. Módulos de kernel

El kernel de Linux permite ser ampliado en **runtime** con módulos. Los módulos son objetos de código compilado que pueden ser insertados en runtime al kernel, siendo linkeados contra el kernel al momento de ser insertados. De esta manera puede ampliarse la funcionalidad del kernel en runtime, sin tener que incluir todo el código en el binario original.

7.6.3. Compilando un módulo de kernel

A continuación compilaremos y probaremos un módulo de kernel muy simple. Este módulo simplemente escribe en la consola "Hola kernel!" al ser insertado y "Chau, kernel." al ser removido. Cree el siguiente módulo en el archivo `/home/<usuario>/tp/hello.c`.

7.6.4. hello.c

Para compilar este código deberá construir una **Makefile**. Este archivo Makefile es utilizado luego por el comando `make` para compilar el módulo con las opciones correctas. En el mismo directorio donde se encuentra `hello.c` cree un archivo `Makefile` conteniendo:

7.6.5. Makefile

Luego ejecute:

```
make
```

Para compilar el módulo. Finalmente, pruebe insertar el módulo usando:

```
insmod hello.ko
```

Debería ver el mensaje "Hola kernel!" en la consola. Quite el módulo usando:

```
rmmod hello.ko
```

Debería ver el mensaje "Chau, kernel".

7.6.6. Un módulo propio

Escriba un módulo de kernel que permita controlar los LEDs del teclado **sin necesidad de escribir un programa en lenguaje C**. El módulo deberá permitir prender o apagar cada LED usando simplemente comandos del shell.

Dado que la máquina virtual provista por VirtualBox no muestra el estado de los LEDs, la cátedra provee abajo un programa que lo hace. Compile utilizando:

```
gcc -o check_kbleds check_kbleds.c
```

7.6.7. check_kbleds.c

Sugerencia Haga que su módulo cree un archivo en `/proc` y que las escrituras a ese archivo controlen los LEDs utilizando la IOCTL `KDSETLED` de la consola de Linux.

Referencias

[BUR97] Richard L. Burden, J. Douglas Faires - Numerical Analysis 6th Edition