Universidad Tecnológica Nacional - Facultad Regional Santa Fe

AÑO 2022

SISTEMAS OPERATIVOS
TRABAJO PRÁCTICO 2:
"Llamadas al Sistema"

Objetivos del práctico

Al terminar este trabajo Ud. habrá aprendido a:

1. Utilizar convenientemente algunos de los principales comandos del SO LINUX.
2. Configurar dispositivos de almacenamiento secundario (disketteras/imágenes)
3. Conocer editores y familiarizarse con el que mejor se adapte a su requerimiento.
4. Generar archivos ejecutables (archivos interpretados-scripts).

Herramientas necesarias:

Para resolver los ejercicios propuestos necesitará:

1. Una PC con SO Windows con el Software de Virtualización VMWARE.
2. El material de Campus de la Cátedra.

Comentado [U1]: Poner Windows en lugar de las versiones

Comentado [U2]: Software de virtualización en lugar de emulador

Fuentes de Información sugeridas

Encontrará información útil en:

- Páginas de manual de LINUX
- El material provisto por la Cátedra.
- Guía de clases de laboratorio: Uso de Comandos Básicos de Linux - El Shell -
- http://www.debian.org/doc/manuals/reference/ch-tutorial.es.html
- http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
- Kernighan, Brian W.; Pike, Rob. (1984). The Unix Programming Enviroment, Prentice Hall.
- Tutorial BASH de MacProgramadores
- Newham, Cameron. (2005). Learning the Bash Shell, Third Edition, O'Reilly Media
- Shotts, William E., Jr - The Linux Command Line http://linuxcommand.org/tlcl.php
- http://tldp.org/LDP/abs/html/tests.html

Requisitos de Entrega

Lugar y Fecha de entrega:

1. La fecha de entrega para este práctico será informada por el CAMPUS en el momento de publicar el TP.
2. Los trabajos deben ser entregados vía CAMPUS con el formato: **"TP2 - GRUPO XX"** (XX es el número que identifica al grupo).
3. No se aceptarán trabajos incompletos.

Comentado [U3]: No están entregando por CAMPUS?

Formato de Entrega.

Deberá enviar dos archivos con la resolución del trabajo:

1. La imagen de un diskette en formato ext2 conteniendo los scripts.
2. El segundo, es un archivo de texto. Deberá reunir las siguientes características:
    1. Secciones del documento (Todas obligatorias):
        1. **Carátula de presentación:** Debe incluir OBLIGATORIAMENTE:
            1. Asignatura
            2. Número y Descripción del trabajo práctico
            3. Año y Cuatrimestre de Cursado
            4. Identificación del Grupo
            5. Nombres, Apellidos y direcciones de correo electrónico de TODOS los Integrantes del grupo

        2- Sección Principal: Aquí debe incluirse la resolución de cada uno de los problemas planteados. **El código de los problemas deberá estar en formato de TEXTO (no captura de imagen)**. Para cada respuesta debe indicarse OBLIGATORIAMENTE, el número y título del problema al que corresponde tal como aparece en el enunciado.

        3- Sección de Descargos: Aquí debe incluirse cualquier comentario que deba tenerse en cuenta para la corrección del práctico. Use esta sección para indicar cosas como:

        - Qué no pudo resolver alguno de los problemas
        - Qué no pudo resolver COMPLETAMENTE alguno de los problemas.
        - Qué no está seguro si el problema está resuelto correctamente.

        Comentar los problemas en esta sección es la única forma de obtener puntaje parcial para un ítem que no está bien resuelto. Si se encuentra un problema no resuelto o resuelto de manera INCOMPLETA y eso no está comentado en esta sección, perderá puntos adicionales (no sólo le descontaremos puntos por el error sino también por no avisarnos). Si no tiene ningún comentario, deje esta sección en blanco.

Penalizaciones.

Los prácticos entregados en fechas posteriores al límite fijado, tendrán una quita de puntos. Para ver el método empleado para restar puntos consulte en el Campus.

Cambios al enunciado del práctico, fechas de entrega, etc.

Cualquier cambio en los enunciados, fechas de entrega, etc. será informado utilizando dos métodos:

1. El campus virtual.
2. La lista de correos.

El alumno no puede alegar que no estaba al tanto de los cambios si esos cambios fueron anunciados utilizando alguno de los dos métodos.
SUGERENCIA: Consulte frecuentemente las novedades del Curso en el Campus Virtual y asegúrese de que ha sido incorporado a la lista de correos.

Honestidad académica:

Está bien hablar entre los grupos acerca de cómo resolver problemas, pero los grupos son de hasta 3 integrantes.

No entregue el trabajo de otras personas como propio. Tampoco entregue trabajos publicados en Internet como propios sin citar las fuentes.
Cualquier trabajo, porción de trabajo o texto sin la cita correspondiente es plagio.

Cada grupo debe mantener su código para sí mismo, si su proyecto es copiado, puede ser difícil determinar quién es el verdadero autor.
Cualquier ayuda que reciba deberá documentarla como un comentario al inicio del programa. Por ejemplo, si encuentra una solución a un ejercicio en un texto o manual, debería citar la fuente. Una razonable ayuda, no afectará la aprobación de los trabajos pero fallas al citar las fuentes o la ausencia de las mismas es fraude.

Queda debidamente aclarado, que los trabajos son de autoría, desarrollo y elaboración propia y no de un tercero.

Por último, la Cátedra podrá solicitar revisar el Trabajo Práctico con coloquio presencial con los alumnos del grupo en caso de considerarlo.

ARGUMENTOS POR LÍNEA DE COMANDO (ARGC – ARGV – OPCIONES)

EJERCICIO 1 (20 Pts)
Deberá generar un programa en C que le permitirá manejar opciones y argumentos a través de la línea de comandos.
Sintaxis de ejecución (suponiendo que se lo compila con el nombre **ej1**):
ej1 [-cl] [-s nombre_archivo]
opciones:
- **c**: indicará la cantidad de argumentos con el que fue llamado ej1 (incluyendo también el primer argumento reservado para el nombre del programa)

- **l**: listará cada argumento indicando el número que representa.

- **s**: almacenará en el archivo: nombre_archivo los argumentos con el que se llama el programa incluyendo el nombre del programa. El archivo deberá ser creado al momento de esta acción ya que debe contener sólo la información correspondiente a la corrida actual.

Si no se especifican opciones, deberá imprimir por pantalla un mensaje con la sintaxis del programa.

Ejemplos de corrida:

```
Corrida 1:
# ./ej1
Uso: ./ej1 [-cl] [-s nombre_archivo]

Corrida 2:

# ./ej1 -c
Cantidad de argumentos: 2
Corrida 3:

# ./ej1 -ls
Argumento: 0- ./ej1.
Argumento: 1- -ls.
./ej1: option requires an argument -- 's'

Corrida 4:

# ./ej1 -lc
Argumento: 0- ./ej1.
Argumento: 1- -lc.
Cantidad de argumentos: 2

Corrida 5:

# ./ej1 -cls salidaej1
Cantidad de argumentos: 3
Argumento: 0- ./ej1.
Argumento: 1- -cls.
Argumento: 2- salidaej1.
```

Comentado [U4]: Para esto deben usar la función getopt() https://man7.org/linux/man-pages/man3/getopt.3.html

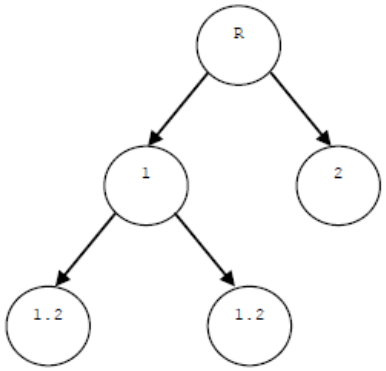
```
Mirando el contenido del archivo pasado como argumento:
# more salidaej1
./ej1-clssalidaej1
```

NOTA:
Para resolver el ejercicio deben usar la función getopt()
https://man7.org/linux/man-pages/man3/getopt.3.html

GESTIÓN DE PROCESOS

EJERCICIO 2: (20 Pts)
Genere un programa en C: ej2.c que mostrará a través del siguiente árbol de jerarquía de procesos, los diferentes resultados para la variable "**VALOR**":

Árbol:



- Nodo **R** (Raíz): corresponde al proceso principal, imprimirá el valor inicial para la variable. Este valor será recibido como argumento por línea de comando.
- Nodo **1**: es el primer hijo del proceso principal, que a su vez será raíz del subárbol 1. Imprimirá el incremento en 100 de la variable VALOR.
  - Nodo **1.1**: corresponde al primer hijo del Nodo 1 e imprimirá el doble de la variable VALOR calculado en el Nodo 1.
  - Nodo **1.2**: corresponde al segundo hijo del Nodo 1 e imprimirá la mitad de la variable VALOR. calculado en el Nodo 1.
- Nodo **2**: es el segundo hijo del proceso principal, mostrará el decremento en 100 de la variable VALOR.

En cada resultado que se muestre se deberá indicar el "nodo" en que se está realizando el cálculo, además de la identificación del proceso. Para los procesos hijos también se deberá identificar al padre.

La variable VALOR sólo puede definirse una sola vez.
Sintaxis de ejecución (suponiendo que se lo compila con el nombre **ej2**):

ej2 nro
Ejemplo de corrida:

```
# ./ej2 345
NODO R - VALOR = 345
ID proceso raiz: 2497
NODO 1 - VALOR = 445
ID NODO 1: 2498 - ID padre NODO 1(NODO R): 2497
NODO 1.1 - VALOR = 890
ID NODO 1.1: 2499 - ID padre NODO 1.1(NODO 1): 2498
NODO 1.2 - VALOR = 222
```

```
ID NODO 1.2: 2500 - ID padre NODO 1.2(NODO 1): 2498
NODO 2 - VALOR = 245
ID NODO 2: 2501 - ID padre NODO 2(NODO R): 2497
```

NOTA:
Para resolver el ejercicio deben usar la función fork() / getpid()/ getppid()
https://man7.org/linux/man-pages/man2/fork.2.html
https://man7.org/linux/man-pages/man2/getppid.2.html

EJERCICIO 3: (10 Pts)
Deberá generar un programa que calcula el factorial del siguiente rango de numeración: 1 a 10.
El programa se llamará factorial.c. Antes de finalizar y luego de realizado el cálculo en caso de que se haya recibido un número valido, deberá imprimir el número de su identificación de proceso. En caso de que no sea un número válido mostrará un mensaje de error y también deberá imprimir el número de ID.
Este programa (factorial) será invocado desde ej3.c reemplazándolo (ej3 será sustituido por factorial). Antes de invocar al programa factorial, ej3 deberá mostrar su propio número de identificación de proceso. En caso de error se deberá indicar que factorial no ha podido ejecutarse.
Al finalizar la corrida, deberá incluir en el archivo de texto de resolución del TP una breve conclusión sobre los números de procesos que fueron identificados.
Sintaxis de ejecución (suponiendo que se lo compila con el nombre **ej3**):

ej3 número
Ejemplo de corrida:

```
# ./ej3 367
Id proceso que invoca: XXXX
Número fuera de rango: 367
Id de Prg-Factorial: XXXX

# ./ej3 3
Id proceso que invoca: XXXX
Factorial de 3 = 6
Id de Prg-Factorial: XXXX

# ./ej3 10
Id proceso que invoca: XXXX
Factorial de 10 = 3628800
Id de Prg-Factorial: XXXX

# ./ej3
Id proceso que invoca: XXXX
Error. Usar ./Nombre_pgr_invoca [número]
Id de Prg-Factorial: XXXX
```

NOTA: los números que identifican a los procesos fueron reemplazados por XXXX en el ejemplo de corrida. Al momento en que presenten la solución del ejercicio deberán aparecer los números reales y expresar la conclusión (la conclusión solamente debe ir en el archivo de texto de resolución del TP).

Para resolver el ejercicio se sugiere la utilización de las funciones execv() para el calculo.

GESTIÓN DE SEÑALES

EJERCICIO 4: (30 Pts)
A los huéspedes del hotel se les ofrece entre otros tantos servicios, el desayuno buffete. Una de las principales preferencias elegidas por los visitantes son las tostadas para poder acompañar con una gran variedad de mermeladas regionales de diferentes sabores.
Deberá simular el comportamiento de la tostadora que tiene ciertas particularidades, teniendo en cuenta las siguientes condiciones:
- 1 de cada 5 huéspedes elige pan negro para tostar.
- Cada tostada de pan negro demora, para la graduación de temperatura ya estipulada: 30 segundos (representar en la simulación por 1 seg) en tostarse, mientras que la de pan blanco tarda 45 segundos (representar en la simulación por 2 seg).
- La tostadora se programa de acuerdo a uno u otro tipo de pan.
- Las tostadas se preparan por pares, pero solamente admite un tipo de programación (blanco/negro).

Cada vez que finaliza la tostadora de tostar el pan, emite una alarma indicando que la tostada está hecha.
Datos para la simulación:
- La tostadora será un proceso a la espera de las señales de los huéspedes (segundo proceso) cuando van a realizar las tostadas (pan blanco –SIGURSR1- / pan negro – SIGURSR2). Por cada señal imprimirá el siguiente mensaje:

"Tostar pan blanco"
"Tostar pan negro"
- Los huéspedes enviarán una señal cuando quieran una tostada:

"Pedido de tostadas de pan blanco"
"Pedido de tostadas de pan negro"
- Se solicita simular la situación promedio de 30 tostadas por día que son solicitadas por los huéspedes.

Sintaxis de ejecución (suponiendo que se lo compila con el nombre tostadora y huesped):

./tostadora
./huesped [pid_tostadora]
Ejemplo de corrida(se muestra la salida parcial hasta 7 huéspedes, pero la resolución deberá mostrar la salida para el total de huéspedes considerando la cantidad de tostadas propuestas):

```
# ./huesped 2611
Pedido de tostada de Pan Blanco. Huésped: 1
Pedido de tostada de Pan Blanco. Huésped: 2
Pedido de tostada de Pan Blanco. Huésped: 3
Pedido de tostada de Pan Blanco. Huésped: 4
Pedido de tostada de Pan Negro. Huésped: 5
Pedido de tostada de Pan Blanco. Huésped: 6
Pedido de tostada de Pan Blanco. Huésped: 7
```

```
# ./tostadora
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (1)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (2)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (3)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (4)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan negro.
Tostadas de pan negro. (5)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (6)
Tostadas listas.
Tostadora libre, esperando pan.
Deseo tostadas de pan blanco.
Tostadas de pan blanco. (7)
Tostadas listas.
Tostadora libre, esperando pan.
```

NOTA:
Para resolver el ejercicio se sugiere la investigación y utilización de las señales SIGUSR1 / SIGUSR2 / SIGALARM
https://man7.org/linux/man-pages/man7/signal.7.html

EJERCICIO 5: (10 Pts)

Generar un programa / proceso llamado **controlc** donde se generen 2 conjuntos de iteraciones de 1 a 100.000; mostrando un mensaje de salidas cada 1000 iteraciones.
Durante el 1er conjunto NO podrá cancelarse el proceso por la combinación de teclas Ctrl+C pero "SI" puede ser posible durante la visualización del 2do conjunto de Iteraciones.

Ejemplos de corrida:

```
Corrida 1:
# ./controlc

Iteración INICIADA.  Presionar Ctrl-C NO tiene efecto....
Iteración 1000
Iteración 2000
Iteración 3000
….
```

```
Iteración 99000
Iteración 100000
Computation is done.

REINICIO de la Iteración.  Presionar Ctrl-C AHORA tiene efecto...
Iteración 1000
Iteración 2000
Iteración 3000
^C
#
```

NOTA:
Para resolver el ejercicio se sugiere la investigación y utilización de señales
https://man7.org/linux/man-pages/man2/signal.2.html

INFORMACION DEL SISTEMA.

EJERCICIO 6: (10 Pts)
Generar un programa llamado **infosistema** donde se obtenga información relacionada al Sistema. (Tipo de Sistema / Nombre del Equipo / Version del Kernel / Version del S.O. / Arquitectura)
.

Ejemplos de corrida:

```
Corrida 1:
# ./infosistema

Tipo de Sistema: Linux
Nombre del Equipo: so2011
Version del Kernel: 2.6.30-2-686
Version del S.O.: #1 SMP Sat Sep 26 01:16:22 UTC 2009
Arquitectura: i686
```

NOTA:
Para resolver el ejercicio deben usar la función uname
https://man7.org/linux/man-pages/man2/uname.2.html