SISTEMAS OPERATIVOS
TRABAJO PRÁCTICO 2:
"Llamadas al sistema en LINUX"

Objetivos del práctico

Al terminar este trabajo Ud. habrá aprendido a:

1. Utilizar algunas de las llamadas al sistema en Linux.
2. Reconocer las librerías que hacen posible el uso de las llamadas al sistema.
3. Escribir un programa en C que utilice las llamadas al sistema en Linux.

Herramientas necesarias:

Para resolver los ejercicios propuestos necesitará:

1. Una PC con SO XP/VISTA/W7/W8/W10 con el emulador VMWARE.
2. El material proporcionado por la Cátedra.

Fuentes de Información sugeridas

Encontrará información útil en:

- Página del campus de la cátedra. Repositorio:
  - Guía de programación en C.
  - Guía de Laboratorio SC.
- Sugeridos: Linux for Programmers and Users – The Linux Programming Interface)
- Páginas man de LINUX.
- Sistemas Operativos Modernos 3° edición – Andrew S. Tanenbaum.

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 e-mail a la dirección de correo: sistemasoperativosutnsantafe@gmail.com en el asunto deberá indicar: **"TP2 - GRUPO XX"** (XX es el número que identifica al grupo).
3 No se aceptarán trabajos incompletos.

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):
      1.1. **Carátula de presentación:** Debe incluir OBLIGATORIAMENTE:
      1.1.1. Asignatura
      1.1.2. Número y Descripción del trabajo práctico
      1.1.3. Año y Cuatrimestre de Cursado
      1.1.4. Identificación del Grupo
      1.1.5. Nombres, Apellidos y direcciones de correo electrónico de TODOS los Integrantes del grupo

1.2. **Sección Principal:** Aquí debe incluirse la resolución de cada uno de los problemas planteados. Para cada respuesta debe indicarse OBLIGATORIAMENTE, el número y título del problema al que corresponde tal como aparece en el enunciado.
1.3. **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.

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.

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

EJERCICIO 1.

Tendrá que generar el siguiente programa:

opciones.c que realizará las acciones que se indican a continuación, a partir de los argumentos que reciba por línea de comandos.

Sintaxis de ejecución (suponiendo que se lo compila con el nombre **opciones**):

opciones [lv] –p <pos>

-l: mostrará la longitud de la cadena del nombre del ejecutable.

-v: mostrará por salida estándar las vocales contenidas en el nombre del ejecutable y la posición que ocupan en dicha cadena.

-p: mostrará por salida estándar la vocal correspondiente a la posición que se recibe como argumento. En caso de que en dicha posición no haya una vocal, deberá mostrar un mensaje aclaratorio.

Por opción incorrecta, deberá mostrar el mensaje de error correspondiente.

Independientemente de las opciones referencias, al final el programa deberá mostrar la cantidad de argumentos con el que fue llamado y el listado de dichos argumentos.

Ejemplos de corrida (se muestran algunos casos):

```
# ./opciones
Cantidad de argumentos: 1
Argumento[0]: ./opciones

# ./opciones -lv -p 2
Nombre del Ejecutable: ./opciones - Longitud:10

Vocal [2] = o
Vocal [5] = i
Vocal [6] = o
Vocal [8] = e

Posición vocal [2] = o

Cantidad de argumentos: 4
Argumento[0]: ./opciones
Argumento[1]: -lv
Argumento[2]: -p
Argumento[3]: 2

# ./opciones -p 9
No es vocal[9]: s

Cantidad de argumentos: 3
Argumento[0]: ./opciones
Argumento[1]: -p
Argumento[2]: 9
```

NOTA: para resolver el ejercicio **deberá** trabajar con la función "**getopt**" para el manejo de opciones por línea de comandos. Consulte la sección 3 del Manual para dicha función.

GESTIÓN DE PROCESOS

EJERCICIO 2.

Jack es un famoso actor y debe realizar una escena de riesgo en la película que está protagonizando, para lo que utiliza un doble. El reemplazo del actor por su doble se realiza de tal manera que no se detecta el cambio y es como si siempre hubiera sido el mismo actor.

Deberá simular la situación anterior:

- Generando dos programas: jack.c y doble.c
- El proceso **doble** es quien realizará la escena de riesgo, la misma será simulada calculando la longitud de una cadena que se recibirá como argumento (sólo en caso de que esta acción contenga un error se podrá ver un mensaje que diga "Toma siguiente, no pudo correrse la escena". Cuando el doble ejecuta la acción deberá mostrar el el siguiente mensaje: "Yo -ID- lo he logrado"
- El proceso **Jack**, antes de llamar al doble para que haga la escena de riesgo (ejecutando **doble)**, mostrará un mensaje que diga: "Soy el Gran Jack –ID- ".

Sintaxis de ejecución (suponiendo que se lo compila con el nombre **jack**):
jack [cadena_acción]

Ejemplo de corrida:

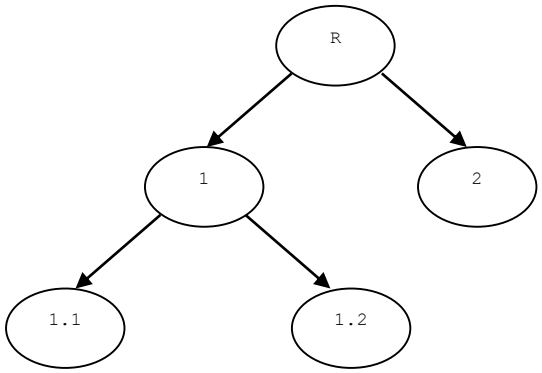
./**jack** escalar
Soy el gran Jack: 2175
Escena de riesgo(Long cadena): escalar = 7
Yo 2175 lo he logrado

NOTA: Deberá utilizar las llamadas al sistema para gestión de procesos que permiten ejecutar y sustituir la imagen de un proceso e identificar procesos.

EJERCICIO 3.

Genere un programa en C: **valorArbol.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 **valorArbol**):

valorArbol nro

Ejemplo de corrida:

./**valorArbol** 1000

NODO R - VALOR = 1000

ID proceso principal: 2443

NODO 1 - VALOR = 1100

ID NODO 1: 2444 - ID padre NODO 1(NODO R): 2443

NODO 1.1 - VALOR = 2200

ID NODO 1.1: 2445 - ID padre NODO 1.1(NODO 1): 2444

NODO 1.2 - VALOR = 550

ID NODO 1.2: 2446 - ID padre NODO 1.2(NODO 1): 2444

NODO 2 - VALOR = 900

ID NODO 2: 2447 - ID padre NODO 2(NODO R): 2443

NOTA: para resolver el ejercicio tenga en cuenta usar llamadas para crear procesos e identificarlos.

GESTIÓN DE ARCHIVOS, DIRECTORIOS Y SISTEMA DE ARCHIVOS

EJERCICIO 4 .

Genere el programa **listdir.c** que permitirá listar el contenido de un directorio, tendrá la siguiente sintaxis de ejecución:

listdir [opciones] [pathname]

Opciones:

- l: lista el contenido del directorio.
- t: tipo.

  Listará el contenido del directorio especificando el tipo de cada archivo (Regular, directorio, especial de dispositivo de caracteres, especial de dispositivo de bloques, enlace simbólico, etc.)
- i: número de inodo

Pathname: será el nombre de un directorio, en caso de que sea inválido deberá mostrar el mensaje de error correspondiente. Si no se especifica al momento de la ejecución deberá ejecutarse sobre el directorio actual.

Sintaxis de ejecución (suponiendo que se lo compila con el nombre **listdir**):
listdir [-lti] [dir_name]

Ejemplos de corrida:

```
so2011:/home/SO2020/tp2# ./listdir -lti /home/SO2020/tp1
Listado del directorio /home/SO2020/tp1:
TP12020.pdf
redirecciones.sh
3_entorno.sh
procesos.sh
testArch.sh
```

```
bday.sh

Tipos de archivos del directorio /home/SO2020/tp1:
TP12020.pdf - Regular
redirecciones.sh - Regular
3_entorno.sh - Regular
procesos.sh - Regular
testArch.sh - Regular
bday.sh – Regular

Número de inodo de archivos del directorio /home/SO2020/tp1:
TP12020.pdf - 180418
redirecciones.sh - 180416
3_entorno.sh - 180413
procesos.sh - 180415
testArch.sh - 180417
bday.sh – 180414

so2011:/home/SO2020/tp2# ./listdir -l
Listado del directorio actual:
opciones.c
doble.c
ej1
jack
opciones1.c
```

NOTA: para resolver el ejercicio deberá trabajar con las llamadas al sistema de Gestión de Archivos, Directorios y Sistema de Archivos .

GESTIÓN DE SEÑALES

EJERCICIO 5.

Un joven mago ha llegado a una pequeña aldea a ofrecer su espectáculo.

A pocos días de haber comenzado sus funciones, su público ha ido aumentando notablemente. Principalmente su sala está llena de mamás con sus hijos. Su acto es básicamente hacer aparecer regalos que extrae de su galera totalmente vacía.

Se solicita simular el acto de magia teniendo en cuenta lo siguiente:

- Cada 3 golpes de la varita mágica sobre la galera, comienzan a salir flores o golosinas. (cada golpe se representa con 1 seg).
- Los elementos aparecen en forma aleatoria.
- Cuando sale una golosina, el mago la regala a un niño y si una flor la regala a la mamá.
- El mago no extrae de la galera el próximo regalo hasta no haber entregado el anterior. (Tarda 2 seg para entregar cada elemento, lo que toma fuertemente la galera a modo de señal para extraer la próxima).
- En esta función, el mago regalará 10 obsequios en total.

Sintaxis de ejecución (suponiendo que se lo compila con el nombre galera y mago):

./galera [pid_mago]

./mago

./**galera** 2594
Es una FLOR: 1
Es una GOLOSINA: 2
Es una GOLOSINA: 3
Es una FLOR: 4
Es una FLOR: 5
Es una GOLOSINA: 6
Es una GOLOSINA: 7
Es una FLOR: 8
Es una GOLOSINA: 9
Es una GOLOSINA: 10

./**mago**
Dama. Esta flor es para Ud. (1)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niño. (2)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niñoo. (3)
Ha salido un obsequio para ustedes.
Dama. Esta flor es para Ud. (4)
Ha salido un obsequio para ustedes.
Dama. Esta flor es para Ud. (5)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niño. (6)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niño. (7)
Ha salido un obsequio para ustedes.
Dama. Esta flor es para Ud. (8)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niño. (9)
Ha salido un obsequio para ustedes.
Esta golosina es para vos Niño. (10)

NOTA: para resolver el ejercicio deberá utilizar las llamadas al sistema correspondientes al manejo de señales.

REDIRECCIÓN DE ENTRADA Y SALIDA.

EJERCICIO 6.

Una agencia encargada de alarmas tiene un mecanismo automático para generar claves a sus clientes, usuarios del sistema de alarmas.

Deberá modelar esta situación teniendo en cuenta lo siguiente:

USUARIOS → CLAVES → ARCHIVO DE CLAVES

Generando los siguientes programas:

Un programa llamado **usuario.c** recibirá los nombres de los usuarios a dar de alta en un archivo llamado nombres.txt. Por cada nombre diferente se generará un usuario con el siguiente formato: la letra u + nombre.

El programa deberá ejecutarse de la siguiente manera:
`./usuario < nombres.txt`

Ejemplo de salida:
```
#more nombres.txt | ./usuario
```

Salida:
```
uLuis
uMaría
uDana
```

Un programa llamado **clave.c** que generará la clave con un número fijo de 4 dígitos (por ejemplo 3341) concatenado al nombre.

Deberá ejecutarse de la siguiente manera:
`./clave < nombres.txt`

Ejemplo de salida:
```
#more nombres.txt | ./clave
```

Salida:
```
Luis3341
María3341
Dana3341
```

Un programa llamado **habilitar.c** que será lo mismo que ejecutar :

#./usuario < nombres.txt |./clave > usuarios.txt

Guardando la salida en el archivo **usuarios.txt.** Es decir, generará los procesos correspondientes con las redirecciones adecuadas ejecutando clave y usuario tal cual como fueron generados anteriormente.

NOTA: para resolver el ejercicio deberá trabajar con las llamadas al sistema de Gestión de Procesos adecuadas y las necesarias para conectar + redireccionar la entra y la salida.