2015

Facultad de Ingeniería y Tecnología Informática

Universidad de Belgrano

SISTEMAS OPERATIVOS
GUIA DE TRABAJOS PRACTICOS
SISTEMAS OPERATIVOS

Ing. en Informática (502-2015)

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**1**
Tema	**Utilidades de Desarrollo – Instalación y Configuración de SOp.**
Tipo de Práctica	**Formación Experimental (P1) – ejercicios 1 a 8**

1. Objetivos:

Con la resolución de los siguientes ejercicios se busca:

- ❖ Familiarizar al alumno con herramientas esenciales para el conocimiento de SOp.
- ❖ Proveer a la base de conocimiento experimental que utilizará, en extenso, en los siguientes prácticos.

2. Introducción de contexto

El grupo de alumnos deberá realizar la instalación y configuración de una Plataforma Operativa Linux, en Distribución y Versión a elección del grupo de trabajo. Se recomienda el uso de Linux Ubuntu 12,04, Debian 6.0, Fedora 16, LinuxMint Cinnamon, o Mandriva 12, DSL 4.3. O cualquiera que se justifique su instalación.
El TP constará de dos etapas, una previa a la instalación y otra posterior de respuesta a los ejercicios propuestos.
La instalación puede hacerse:
a. Instalación real en HDD privados, determinados por el grupo.
b. Instalación a través de Virtualización. (Recomendada).
c. Instalación a través de wubi de Ubuntu. Esta tiene la limitación que la única distro para hacerlo es Ubuntu,
No hay una configuración pre-establecida de Hardware, la elección es libre.

3. Ejercicios previos a la instalación

1. Responda las siguientes preguntas sobre Particiones:
   a. Defina el concepto de Partición. Cuales tipo de particiones conoce. Ventajas y Desventajas de este sistema.
   b. ¿Como se identifican las particiones en Linux? ¿Y en Windows Server? (Independiente de la controladora del Disco)
   c. ¿Cuantas particiones son necesarias como mínimo para instalar Linux? ¿Cuáles? Porque se utiliza este sistema? Conoce algún otro?
   d. De ejemplos de distintos casos de particionamiento dependiendo del tipo de tarea que se deba realizar en su sistema operativo
   e. ¿Qué tipo de software para realizar particiones existe? Menciónelos y compare. Y Uds. cual utilizarían. Justifique las respuestas.

2. Sobre Distribuciones Linux:
   a. Porque hay distintos tipos de distribuciones de Linux disponibles.

b. En qué se diferencia una distribución de otra?
c. Cuáles de las distribuciones mencionadas anteriormente son más estables?

3. Responda lo siguiente sobre Kernel.
   a. Qué es?
   b. Cuales son sus funciones?
   c. Que versiones existen? ¿Cómo se las diferencian?
   d. Es posible tener más de un Kernel de Linux.
   e. Cual es el directorio que corresponde al Núcleo – Kernel dentro del Sistema de Archivo?
4. Desarrolle una hoja de Configuración previa a la Instalación y posterior con los datos siguientes (Todo esto forma parte de la documentación de su sistema – utilice formatos normalizados)
   a. Datos de la configuración de Hardware seleccionada para la instalación.
   b. Datos del Sistema Operativo Linux seleccionado, Versión y Distribución.
   c. Estado final de la Instalación.

4. Ejercicios posteriores a la Instalación

1. Listado impreso de la configuración de Hardware funcional lograda. Impresión desde la línea de comandos. En modo carácter.
2. Listado impreso del archivo /etc/grub.conf
3. Listado impreso del archivo /etc/inittab.
4. Listado impreso del directorio raìz de la propia configuración, utilizando un manejador de archivos de la GUI configurada para su plataforma.
5. Listado impreso de la configuración de la GUI.
6. Creación de los siguientes usuarios: usradm1 y usradm2, los cuales deben tener capacidad de administración de propia sesión, pero no ser administradores de sistema.
7. Listado impreso desde línea de comando de los respectivos archivos de configuración ".profile" (para el caso de shell sh) y/o ¨.BASH¨(para el caso de shell bash) de los usuarios creados en el punto 6, utilizando cualquier editor de texto de su GUI configurada.

5. Ejercicios sobre Compilador gcc

1. Cual es la funcionalidad de cada expresión escrita? Desarrolle ejemplos.
   a. $gcc -E file.in -o file.cpp
   b. $gcc -x cpp.out -c filein.cpp -o file.out
   c. $gcc file.in -o file.out
   d. $gcc file1 file2 file3 -o file.out

2. Que tipo de archivo es un .cpp? Desarrolle ejemplos.

3. Cual es la función que cumplen los siguientes argumentos de gcc?
   -static -ansi -w -g
   Desarrolle ejemplos.

4. Desarrolle un programa que incluya una biblioteca estática, que permita ver por pantalla una serie mensajes requeridos, previamente escritos.

5. Escriba un programa que permita ver cuales procesos se encuentran ejecutándose (o en activo) en memoria. Que la lista se ordene por el tamaño y por la antigüedad de los procesos. Optimícelo hasta el nivel 03, compare el tamaño de los programas, sin optimizar y el optimizado. Desarrolle las conclusiones.

6. El programa desarrollado en el ejercicio 5, sométalo a la acción del debugger. Indique cuales diferencias se presentan al final de su tarea.

7. Al usar el make, cual es la funcionalidad que tienen:
   a. Target.
   b. Dependencias.
   Explique y de ejemplos en cada caso.

8. Crear un target de Instalación, de distribución y de desinstalación; en función del programa desarrollado en el ejercicio 5.

En el caso de realizar Virtualización, hacerlo con VirtualBox.
LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**2**
Tema	**Procesos y Estructuras de Ejecución**
Tipo de Práctica	**Formación Experimental (P1) – ejercicios 1 a 9** **Problemas abiertos de Ingeniería (P2) – ejercicio 10**

1. Objetivos

Los objetivos de este trabajo práctico son los siguientes:

- Aplicar los conceptos básicos de administración de procesos. Utilizando los diversos algoritmos para planificación de procesos.
- Estudiar algunos de los principales system calls para administración de procesos en UNIX (getpid, getppid, getuid, getgid, exit, fork, exec, wait, waitpid, sleep) y sus macros asociadas (WIFEXITED, WEXITSTATUS, WIFSIGNALED, WTERMSIG).
- Usar adecuadamente los principales comandos para monitoreo de procesos: ps, pstree, top, gtop, kpm, kill, etc.

2. Ejercicios

1. Dada la siguiente tabla de procesos en ejecución se solicita que aplique el Algoritmo Round Robin para ejecutar la siguiente secuencia de procesos.

Proceso	Instante de llegada	Tiempo de Servicio	Tiempo de I/O
A	0	5	IRQ final 1C/4C
B	2	3	--
C	5	4	IRQ final 2C/2C
D	7	2	--
E	8	6	--

2. Resolver el siguiente ejercicio según lo explicado a continuación:

PROCESOS	Tllegada	Tservicio	OPERACIONES
A	0	4	***IRQ final 2C/3C***
B	3	5	***IRQ final 4C/5C***
C	5	7	**--**
D	6	2	**--**
E	7	8	***IRQ inicio 4C/3C***
F	9	4	**--**
G	10	3	***IRQ Final 1C/2C***

2.1. Información de Contexto:

- Delay = 1C (Por Overtime del Despachador)
- Quantum = 2C
- Llega una IRQ de I/O por una llamada de ejecución RCP, durante el inicio del Ciclo 14, y tiene una duración de 4C.

2.2. Resolver:

2.2.1. Resolverlo aplicando sistema Round Robin. Desarrollar conclusiones del problema propuesto.

2.2.2. Aplicar el Algoritmo que dado el planteo de tiempos, permita favorecer a los Procesos más largos.

2.2.3. En el caso de aplicar esta tabla para un sistema de corto plazo de tipo no preemtivo. Resolverlo favoreciendo a los procesos cortos. Considerar sólo las tres columnas iniciales, sin operaciones.

3. Completar el siguiente programa en C para que imprima en su salida standard su identificación de proceso (pid), la identificación de proceso de su padre (ppid), la identificación del usuario que lo puso en ejecución (uid) y la de su grupo (gid).

```
# include <sys/types.h>
# include <unistd.h>
int main (void)
{
        printf ("PID: %d \n", getpid ( ) );
        printf ("PPID:  %d \n", .....);
        .....
        .....
        .....
        exit (0);
}
```

4. El código de retorno de un proceso puede obtenerse mediante la variable $? del shell, por ejemplo:

```
$ cd
$ ls .bash_profile
.bash_profile
$ echo $?
0
$ ls archivoinexistente
archivoinexistente: No such file or directory
$ echo $?
1
```

Escriba un programa que devuelva como código de retorno el valor entero, entre 0 y 255, pasado como primer parámetro.

5. Ejecute el siguiente programa y explique su funcionamiento:

```
# include <sys/types.h>
# include <unistd.h>
int main (void)
{
        pid_t pid;

        printf ("Just one process so far \n");
        printf ("Calling fork ....\n");
        pid = fork ( );
        if ( pid = = 0 ) {
             printf ("I´m the child: %d  ", getpid( ));
             printf (" and my parent is : %d \n ", getppid( ));
```

```
        } else if ( pid > 0 )  {
              printf ("I´m the parent: %d  ", getpid( ));
              printf (" and my child is: %d \n ", pid );
        } else
              printf ("fork returned error code, no child \n");
}
```

6. Escriba un programa en gcc que cree dos procesos hijos y estos a su vez creen un nuevo proceso hijo. Después de cada ejecución de fork( ), cada unos de los procesos padres deberá imprimir en salida Standard el pid del hijo recientemente creado.
   Use la Llamada del Sistema sleep( ) para demorar la finalización de los procesos hijos.
   Desde otra terminal verifique la creación y muerte de procesos

7. Escribir un programa usando compilador gcc que cree un proceso hijo (fork( )), espere la finalización del mismo (wait y waitpid) e imprima en la salida standard la siguiente información.
   - Si la terminación fue normal (WIFEXITED) su código de retorno (WEXITSTATUS)
   - Si la terminación fue anormal (WIFSIGNALED) el número de señal que ocasionó su terminación (WTERMSIG).

   El proceso hijo por su parte imprimirá en la salida standard su pid y luego esperará 30 segundos antes de finalizar. Esto le dará a Ud. la posibilidad de cancelarlo mediante la ejecución del comando kill.

8. Compile y ejecute el siguiente programa. Explique su funcionamiento.

```
# include <sys/types.h>
# include <unistd.h>

void fatal (char * s) ;
int main (void)
{
        pid_t pid;
        int     status;

        pid = fork ( );
        if ( pid = = 0 ) {
        execl ("/bin/ls", "ls", "-l", NULL);
        fatal ("execl failed");
        } else if ( pid > 0 )  {
         wait (&status);
            printf ("ls completed\n" );
        } else
            fatal ("fork failed");
}

void fatal (char * s)
{
        perror (s);
        exit (1);
}
```

9. Modifique el programa anterior para que permita la creación de un proceso hijo, y la ejecución del programa ingresado como parámetro de la línea de comandos.

10. Utilizando la tabla de procesos que figura en el ejercicio 2. Desarrollar un programa usando el compilador gcc, que permita resolver el problema de planificación planteado utilizando Round Robin.
    Los datos de ingreso deben estar conformados por los datos de la tabla del ejercicio 2. Los datos de salida deben dar la secuencia de finalización y el diagrama de Gantt correspondiente, mostrando la secuencia total de resolución.

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**3**
Tema	**Sincronización e IPC**
Tipo de Práctica	**Formación Experimental (P1) – ejercicios 1 a 5 y 7 a 8** **Problemas abiertos de Ingeniería (P2) – ejercicio 6**

1. Objetivos

Los objetivos de este trabajo práctico son los siguientes:

- Resolver ejercicios de aplicación de diversas soluciones a problemas de sincronización presentados cuando se busca acceder a un recurso.
- Solucionar diferentes problemas presentados con el uso de IPC.
- Trabajar con los antiguos IPC System V e IPC Posix.

2. Ejercicios

1. Resolver el siguiente ejercicio según lo indiquen las condiciones de contexto:

PROCESOS	Tllegada	Tservicio	OPERACIONES
A	0	20	***IRQ final 5C/10C***
B	1	15	***IRQ final 5C/12C***
C	2	18	***IRQ final 4C/8C***

1.1. Información de Contexto:

a. Delay = 1C (Por Overtime del Despachador)
b. Quantum = 2C
c. Ambas IRQs internas deben localizar en la dirección 48Fh la Variable prueba, que en Tcero tiene un valor de 4.
d. La IRQ del Proceso A, debe cambiar el valor de la variable prueba=8.
e. La IRQ del Proceso B, debe cambiar el valor de la variable prueba=12.
f. La IRQ del Proceso C, nuevamente cambia el valor de prueba=16

1.2. Resolver:

Usando una planificación de corto plazo con Round Robin resolver la ejecución de los procesos en pugna. Aplicando el Algoritmo de Dekker o Peterson (el que mejor se adapte) asegurar la Sección Crítica de los tres procesos y determinar cual será el valor en el ciclo final de ejecución, de la variable que se localiza en la dirección 48Fh.

2. Explique como el semáforo **s** asegura la Exclusión Mutua y en que lugar lo hace:

```
/* Programa de Exclusión Mutua*/
const int n= /* Numero de Procesos */
semáforo s=1;
void P(int i)
{
    while (cierto)
    {
      wait(s)
      /*  Sección Crítica */
      signal(s)
      /*resto*/
    }
}
void main( )
{
    parbegin(P(1), P(2), …, P(n));
}
```

3. Determinar la secuencia de acceso al recurso según los datos de la tabla de procesos siguientes, utilizando Semáforos Duro.

Proceso	Instante de llegada	Tiempo de Servicio
A	0	3
B	2	4
C	4	5
D	5	2

4. El proceso A y B que tienen el mismo padre, necesitan enviarse una información para continuar con sus operaciones. Para ello deberá crear y proceder a escribir un pipe, a continuación leerlo y cerrarlo. Por último devolver los descriptores correspondientes y mostrando en pantalla el mensaje que fue leído por B, enviado por A. Mensaje: “We are the champions”
5. Crear un FIFO que permita ver la evolución de su ejecución con sus diversas salidas, al ser accedido por diversos programas.
6. Tengo el Proceso Alfa, el cual debe completar una operación Polinómica, cuyo resultado es la sumatoria de m miembros. Dicho resultado le servirá al proceso Beta para usarlo como semilla para calcular y emitir un número que cumpla criterios de Primalidad. Dicha clave se utiliza como base del Producto del Algoritmo RSA. Realizar por Segmentos de Memoria compartida.
7. Conformar una cola de mensajes que permita listar 7 mensajes, y que estos puedan visualizarse en stdout. Leer en forma directa el 4 y 6 mensaje, mostrando el resultado en stdout.
8. Desarrollar un programa que permita sincronizar el acceso a un recurso compartido (determinado por el programador) dada por la competencia de los procesos A, B y C.

Deberá mostrarse cuando accede uno de ellos al recurso y los restantes se bloquean a la espera de la liberación del recurso. Terminar eliminando el o los semáforos usados.

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**4**
Tema	**Señales y Sistemas de Llamadas**
Tipo de Práctica	**Formación Experimental (P1) – ejercicios 1 a 5** **Problemas abiertos de Ingeniería (P2) – ejercicio 8** **Prácticas de Proyecto y diseño de Sistemas Informáticos (P3) – ejercicio 6 y 7**

1. Objetivos

Los objetivos de este trabajo práctico son los siguientes:

- ❖ Aplicar los conceptos de señales y mecanismos básicos de IPC aplicados a señales e interrupciones.
- ❖ Estudiar las principales Llamadas del Sistema (system calls) para manipulación de señales.

2. Ejercicios

1. Es sabido que al ejecutar un comando o un programa desde la consola, mediante la combinación de teclas ctrl-c se puede terminar ese proceso. Se pide desarrollar un programa que capture esa señal y redefina su comportamiento, imprimiendo algo por la salida estándar. Al presionar ENTER se debe lanzar la señal SIGQUIT para terminar el proceso.

Nota: La señal lanzada al presionar crtl-c es SIGINT. Para que un proceso se envíe a si mismo una señal se puede utilizar la función raise( ).

2. Para determinar el comportamiento de alarm( ), crear un programa que establezca una alarma para dentro de diez (10) segundos, mientras imprime números consecutivos indefinidamente. Sacar conclusiones.

3. Redefina el ejercicio anterior de manera tal que al cumplirse el tiempo de alarma, el programa se detenga durante cinco (15) segundos y luego continúe imprimiendo.

Nota: alarm( ) envía la señal SIGALRM. Utilizar la función pause( ) para hacer la detención.

4. Crear un programa que bloquee las señales SIGTERM y SIGALRM utilizando una máscara de señales que las contenga. Para comprobarlo, desde el main( ), establecer un alarma de cinco (5) segundos y lanzar SIGTERM.

5. Crear un programa que bloquee la señal SIGINT durante setenta y cinco (75) segundos, y luego la desbloquee.

6. Crear un programa que bloquee las señales SIGINT y SIGKILL, luego lanzar SIGINT y SIGKILL. Antes de ejecutar el programa, definir cual debería ser el comportamiento y contrastarlo con el comportamiento de ejecución.

7. Crear un programa que bloquee las señales SIGTERM y SIGALRM. Luego, lanzar cualquiera de las dos, si la señal bloqueada fue SIGTERM, imprimir ?SIGTERM?, si la señal fue SIGALRM, desbloquearla y establecer una nuevamente alarma de cinco (5) segundos.

8. Desarrollar un programa que lea de un archivo dos matrices y luego las multiplique.
   El proceso de multiplicación debe ser llevado a cabo por tantos threads como filas tenga la matriz. El esquema de administración de threads debe ser del tipo “productor-consumidor”, donde el thread principal del programa es el productor y los threads de multiplicación son los consumidores.
   El thread principal debe mantener a todos los threads ocupados mientras queden filas por multiplicar.
   La matriz resultante debe ser almacenada en un archivo, en forma ordenada.

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**5**
Tema	**Sistemas de Archivos**
Tipo de Práctica	**Formación Experimental (P1) – ejercicios 1 a 4** **Problemas abiertos de Ingeniería (P2) – ejercicio 5** **Prácticas de Proyecto y diseño de Sistemas Informáticos (P3) – ejercicio 5**

1. Objetivos

Los objetivos de este trabajo práctico son los siguientes:

- ❖ Aplicar los conceptos de manejo de archivos y sus seteos.
- ❖ Estudiar las principales funciones y librerías de Linux para manejar archivos.

2. Ejercicios

1. Desarrolle un programa que llame a umask para establecer una máscara umask de mayor restricción de acceso a un archivo. Y explique su funcionamiento.

2. Desarrolle un programa que permita abrir y cerrar un archivo llamado “soyde502 o 501”. Use las librerías unistd.h y fcntl.h.

3. Desarrolle un programa que contenga un archivo con contenido de un string cualquiera y luego use fchmod para manipular sus permisos de acceso. Y detalle que funciones cumplen chown y fchown.

4. Desarrolle un programa que permita listar el contenido de un directorio cuyo nombre es ingresado desde línea de comandos.

5. Desarrollar un script que funcione bajo Shell bash, que permita administrar las cuotas de disco asociadas a los usuarios. El guión puede llamarse gestión_cuotas que solicite el nombre del usuario y le permita elegir entre tres opciones:
   a. minima, 20Mb.
   b. media, 40Mb.
   c. alta, 100Mb.
   Cuando el usuario nombrado tiene cuota previa, le debe preguntar si quiere modificarla.
   Si la quiere modificar, entonces le ofrece nuevamente las tres opciones nombradas.
   Ver la orden:
   man bash | gtroff –t –e -mandoc -Tascii | col –bx > bash.txt
   Estudiar con detenimiento quota y setquota

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**6**
Tema	**Sistemas de Entrada/Salida y Memoria**
Tipo de Práctica	**Formación Experimental (P1) – ejercicio 1 y 2** **Problemas abiertos de Ingeniería (P2) – ejercicio 3** **Prácticas de Proyecto y diseño de Sistemas Informáticos (P3) – ejercicio 4**

1. Objetivo

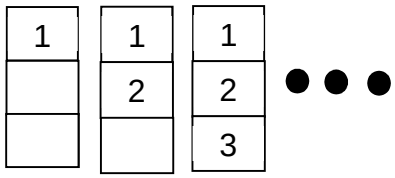
Los objetivos de este trabajo práctico son los siguientes:

- Que el alumno solucione problemas sencillos de aprovechamiento de memoria.
- Realizar un desarrollo que permita manipular la entrada y salida, en modo background.
- Desarrollar aplicaciones comunes del mundo del manejo de puertos de entrada y salida.

2. Ejercicios

1. Dada la siguiente secuencia de carga de páginas, solucionar su reemplazo con el Algoritmo que produzca el menor número de Fallos de página.

   1-2-3-4-5-1-6-7-1-2-3-4-6-7-1-1-2-3-4-5-6-7



2. El archivo Examen.bin es un binario, que es cargado en memoria central para su ejecución. Consta de una extensión de 190 Kb, y esta dividido lógicamente en tres módulos; el Programa Principal que corresponde a ¾ del código total, y dos funciones de complemento, que representan el ¼ restante. Es necesario resolver lo siguiente:
   a. Cuantos procesos conformará el Administrador de Procesos, para ser cargados en la Unidad de Memoria Principal? Se utiliza un sistema de administración por Memoria Virtual por Paginación. En la asignación por procesos, se trata de continuar el concepto de que la división física y lógica serán correspondientes.
   b. Si cada página tiene un tamaño de 8Kb. Cuantas páginas se asignarán por proceso?
   c. Cual será el ordenamiento secuencial de la asignación de páginas? Graficar las correspondientes TMP, considerando que estarán disponibles desde el cuadro de Página número 20.
   d. Se necesita ubicar la siguiente dirección lógica, en la Unidad de Memoria Central (tamaño máximo de la dirección es de 16 bits): **0001001000011111**

Cual es la dirección física correspondiente? Calcular y graficar la correspondiente conversión.

e. Luego de un tiempo de trabajo, cuando ya han ingresado a la UMC otros procesos y se han realizado varios intercambios, el espacio de Memoria asignado a los procesos del archivo Examen.bin se ha reducido a 10 cuadros (distribuidos según la proporción de los procesos, 6-2-2). La CPU hace el pedido de la siguiente secuencia de Páginas:

P5 P19 P20 P21 P20 P24 P21 P20 P19 P5

P1	P2	P3	P4	P5		P19	P20	P22	P23	tz

Utilizar el Algoritmo de reemplazo que permita obtener la menor cantidad de Fallos de Páginas.

3. Desarrollar un programa Daemon para crear un archivo de registro /var/log/lproc.log, que va escribiendo la fecha y hora de los procesos que se encuentran en ejecución, cada cinco minutos. Solo puede ejecutarse en background, y siendo root.

4. Implementación de un manejador de terminal básico. Apuntando principalmente a solucionar los problemas básicos de sincronización. La idea es implementar un servicio básico de lectura de terminal que use el System Call “int leer_caracter( ).
La función nombrada deberá leer un carácter de la terminal y lo devolverá como un resultado de la función. Tenga en cuenta la funcionalidad de getchar( ).

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO

Asignatura	**Sistemas Operativos**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2012**
Ciclo	**3ero**
Cuatrimestre	**1ero**
Trabajo Práctico	**7**
Tema	**Seguridad de los Sistemas Operativos**
Tipo de Práctica	**Formación Experimental (P1) – ejercicio 1, 3, 5** **Problemas abiertos de Ingeniería (P2) – ejercicio 2 y 6** **Prácticas de Proyecto y diseño de Sistemas Informáticos (P3) – ejercicio 4**

1. Objetivos

Los objetivos de este trabajo práctico son los siguientes:

- ❖ Que el alumno solucione problemas sencillos de aprovechamiento de memoria.
- ❖ Realizar un desarrollo que permita manipular la entrada y salida, en modo background.
- ❖ Desarrollar aplicaciones comunes del mundo del manejo de puertos de entrada y salida.

2. Ejercicios

1. ¿Qué significa que la entropía de una clave sea alta? Bajo el punto de vista de la criptografía, ¿es bueno o malo que sea un valor alto y por qué?

2. Alicia (A) desea transmitir un mensaje M a Bernardo (B) dentro de un cuerpo finito Z_n, con n = 27, el número de letras del alfabeto utilizado, el español. El cifrado mediante RSA será letra a letra. Si el grupo y claves que utiliza cada usuario es:

   Alicia (A): $n_a = p \times q = 5 \times 7 = 35$ $\Rightarrow$ Claves: $(n_a, e_a) = (35, 5)$
   Bernardo (B): $n_b = p \times q = 3 \times 11 = 33$ $\Rightarrow$ Claves: $(n_b, e_b) = (33, 3)$

   a) Encontrar las claves privadas de Alicia (A) y Bernardo (B).
   b) Cifrar el mensaje M = HOLA que Alicia (A) envía a Bernardo (B).
   c) Descifrar el criptograma que recibe Bernardo (B).

NOTA 1: Los valores de los caracteres en mod 27 se pueden ver en Vigenère.
NOTA 2: Recordar que (A) envía M a (B) cifrándolo como: $C = M^{eb} \pmod{n_b}$

3. Se desea atacar un criptograma para romper un cifrado partiendo solamente de dicho texto cifrado, ¿qué es lo primero que realizaría un criptoanalista y porqué?

4. Phantomboy es un artista de las violaciones de confianza de los servers, asi ha podido cambiar más de una vez sus notas, aprobar exámenes y vendérselos a Julian ”scriptkill” quien provee a gran parte de la escuela, ganando pingües ganancias. Pero Phamtomboy ha sido expulsado de su newsgroup porque le han dicho que le falta espíritu de grupo y

no quiere donar parte de su tiempo al webmail del grupo. Cual piensa Ud., que se ha enterado por un delator, que es la categoría de habitante del ciberespacio que le cabe a Phantomboy dado que este mismo delator le ha dicho que el próximo objetivo son los Servers que Ud. está administrando?

a. Script – Kidder. Por ello implementaré medidas especiales contra la Ingeniería Social y la localización de Exploits.
b. Cracker. Por ello implementaré un estricto control de las líneas telefónicas, buscando llamadas de origen externo contra los servers.
c. Samurai. Implementaré un sistema de detección de scaneo de puertos TCP/IP.
d. Wannaber. No me preocuparé mucho, porque con lo que tengo implementado le será imposible penetrar.
e. Ninguno de las anteriores opciones.

R:_____

5. Cuales son los objetivos del uso de Denial Of Service ?

6. Desarrollar un programa que bloquea un archivo a nivel de escritura y lectura. Utilizar las funciones unistd.h, errno.h, fcntl.h, Y explicar detalladamente cual será su evolución en la ejecución.

LA NO PRESENTACIÓN DEL TP SEGÚN LAS NORMAS PRE-ESTABLECIDAS DESAPRUEBA AUTOMATICAMENTE EL TRABAJO