Conceptos de Sistemas Operativos (IC)
Introducción a los Sistemas Operativos (LI, LS, APU, ATIC)

Trabajo Práctico N° 2

Objetivo

Comprender conceptos básicos de los sistemas operativos, los procesos y su planificación para la administración de la CPU. Introducir y comprender la administración de memoria, considerando las diferentes posibilidades acordes a las características del hardware, distinguiendo el apoyo de este así como las actividades a realizar para la eficiente administración del recurso por parte del Kernel.

Se recomienda, para la autocorrección de los ejercicios, la utilización del simulador de planificación que se publicará en la plataforma.

Temas incluidos

Definición de SO. Componentes de un SO. Apoyo del Hardware: Modos de ejecución, interrupciones. Llamadas al sistema. Programa y Proceso. Planificación de CPU. Colas de planificación. Context Switch. Creación de procesos. Espacio de direcciones. Direcciones lógicas y físicas. Particiones fijas y dinámicas. Paginación y Segmentación. Fragmentación. MMU. Algoritmos de administración.

1. Responda en forma sintética sobre los siguientes conceptos:
    a. ¿Qué es un Sistema Operativo?
    b. Enumere qué componentes/aspectos del Hardware son necesarios para cumplir los objetivos de un Sistema Operativo.
    c. Enumere componentes de un Sistema Operativo
    d. ¿Que es una llamada al sistema (*system call*)? ¿Cómo es posible implementarlas?
    e. Defina y diferencie Programa y Proceso.
    f. ¿Cuál es la información mínima que el Kernel debe tener sobre un proceso? ¿En qué estructura de datos asociada almacena dicha información?
    g. ¿Qué objetivos persiguen los algoritmos de planificación (scheduling).
    h. ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?
    i. ¿Qué tareas realizan los siguientes módulos de planificación?:
        i. Short Term Scheduler
        ii. Long Term Scheduler
        iii. Medium Term Scheduler
    j. ¿Qué tareas realiza el Dispatcher? ¿Y el Loader?
    k. ¿Qué significa que un proceso sea “CPU Bound” y “I/O Bound”?
    l. ¿Cuáles son los estados posibles por los que puede atravesar un proceso? ¿Qué representa que un proceso se encuentre en los estados

enumerados? Utilizando un diagrama explique las transiciones entre los estados.

m. ¿Cuáles de los schedulers mencionados anteriormente se encargan de las transiciones entre los estados enumerados?
n. Defina Tiempo de retorno (**TR**) y Tiempo de espera (**TE**) para un proceso.
o. Defina Tiempo Promedio de Retorno (**TPR**) y Tiempo promedio de espera (**TPE**) para un lote de procesos.
p. Defina tiempo de respuesta.

2. Para los siguientes algoritmos de scheduling:
   - ➢ FCFS (First Come First Served)
   - ➢ SJF (Shortest Job First)
   - ➢ Round Robin
   - ➢ Prioridades

   a. Explique su funcionamiento mediante un ejemplo.
   b. ¿Alguno de ellos cuentan con parámetros para su funcionamiento? Identifique y enunciarlos
   c. Cual es el más adecuado según los tipos de procesos y/o SO.
   d. Cite ventajas y desventajas de su uso.

3. Dado el siguiente lote de procesos en el que todos arriban al sistema en el instante 0 (cero):

Job	Unidades de CPU
1	7
2	15
3	12
4	4
5	9

   a. Realice los diagramas de Gantt según los siguientes algoritmos scheduling:
      i. FCFS (First Come, First Served)
      ii. SJF (Shortest Job First)
      iii. Round Robin con quantum = 4
   b. Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.
   c. En base a los tiempos calculados compare los diferentes algoritmos.

4. Dado el siguiente lote procesos:

Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4

4	6	5
5	8	2

a. Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:
   i. FCFS (First Come, First Served)
   ii. SJF (Shortest Job First)
   iii. Round Robin con quantum = 1
   iv. Round Robin con quantum = 6
b. Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.
c. En base a los tiempos calculados compare los diferentes algoritmos.
d. En el algoritmo Round Robin, qué conclusión se puede sacar con respecto al valor del quantum.
e. ¿Para el algoritmo Round Robin, en qué casos utilizará un valor de quantum alto y qué ventajas y desventajas obtendría?

5. Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First):
   a. Realice el diagrama de Gantt para este algoritmo según el lote de trabajos del ejercicio 5.
   b. ¿Nota alguna ventaja frente a otros algoritmos?

6. Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 5, donde un menor número indica mayor prioridad:

Job	Prioridad
1	3
2	4
3	2
4	1
5	2

   a. Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes:
      i. No Apropiativa
      ii. Apropiativa
   b. Calcule el TR y TE para cada job así como el TPR y el TPE.
   c. ¿Nota alguna ventaja frente a otros algoritmos? ¿Bajo qué circunstancias lo utilizaría y ante qué situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?

7. Inanición (*Starvation*)
   a. ¿Qué significa?

b. ¿Cuál/es de los algoritmos vistos puede provocarla?
c. ¿Existe alguna técnica que evite la inanición para el/los algoritmos mencionados en el inciso b?

8. Los procesos, durante su ciclo de vida, pueden realizar operaciones de I/O como lecturas o escrituras a disco, cintas, uso de impresoras, etc. El Kernel mantiene para cada dispositivo, que se tiene en el equipo, una cola de procesos que espera por la utilización del mismo (al igual que ocurre con la Cola de Listos y la CPU, ya que la CPU es un dispositivo más).

Cuando un proceso en ejecución realiza una operación de I/O el mismo es expulsado de la CPU y colocado en la cola correspondiente al dispositivo involucrado en la operación.
El Kernel dispone también de un "I/O Scheduling" que administra cada cola de dispositivo a través de algún algoritmo (FCFS, Prioridades, etc.). Si al colocarse un proceso en la cola del dispositivo, la misma se encuentra vacía el mismo será atendido de manera inmediata, caso contrario, deberá esperar a que el Kernel lo seleccione según el algoritmo de scheduling establecido.
Los mecanismos de I/O utilizados hoy en día permiten que la CPU no sea utilizada durante la operación, por lo que el Kernel puede ejecutar otro proceso que se encuentre en espera una vez que el proceso bloqueado por la I/O se coloca en la cola correspondiente.
Cuando el proceso finaliza la operación de I/O el mismo retorna a la cola de listos para competir nuevamente por la utilización de la CPU.
Para los siguientes algoritmos de Scheduling:

- FCFS
- Round Robin con quantum = 2

Y suponiendo que la cola de listos de todos los dispositivos se administra mediante FCFS, realice los diagramas de Gantt según las siguientes situaciones:

a. Suponga que al lote de procesos del ejercicio 5 se agregan las siguientes operaciones de entrada salida:

Job	I/O (Recurso,Instante,Duración)
1	(R1, 2, 1)
2	(R2, 3, 1) (R2, 5, 2)
4	(R3, 1, 2) (R3, 3, 1)

b. Suponga que al lote de procesos del ejercicio 5 se agregan las siguientes operaciones de entrada salida:

Job	I/O (Recurso,Instante,Duración)
1	(R1, 2, 3) (R1, 3, 2)
2	(R2, 3, 2)
3	(R2, 2, 3)

4	(R1, 1, 2)

9. Algunos algoritmos pueden presentar ciertas desventajas cuando en el sistema se cuenta con procesos ligados a CPU y procesos ligados a entrada salida. Analice las mismas para los siguientes algoritmos:
    a. Round Robin
    b. SRTF (Shortest Remaining Time First)

10. Para equiparar la desventaja planteada en el ejercicio 10, se plantea la siguiente modificación al algoritmo:

Algoritmo VRR (Virtual Round Robin): Este algoritmo funciona igual que el Round Robin, con la diferencia que cuando un proceso regresa de una I/O se coloca en una cola auxiliar. Cuando se tiene que tomar el próximo proceso a ejecutar, los procesos que se encuentran en la cola auxiliar tienen prioridad sobre los otros. Cuando se elije un proceso de la cola auxiliar se le otorga el procesador por tantas unidades de tiempo como le faltó ejecutar en su ráfaga de CPU anterior, esto es, se le otorga la CPU por un tiempo que surge entre la diferencia del quantum original y el tiempo usado en la última ráfaga de CPU.

    a. Analice el funcionamiento de este algoritmo mediante un ejemplo. Marque en cada instante en que cola se encuentran los procesos.
    b. Realice el ejercicio 9)a) nuevamente considerando este algoritmo, con un quantum de 2 unidades

11. Suponga que un Kernel utiliza un algoritmo de VRR para planificar sus procesos. Para ello, el quantum es representado por un contador, que es decrementado en 1 unidad cada vez que ocurre una interrupción de reloj. ¿Bajo este esquema, puede suceder que el quantum de un proceso nunca llegue a 0 (cero)? Justifique su respuesta.

12. El algoritmo SJF (y SRTF) tiene como problema su implementación, dada la dificultad de conocer la duración de la próxima ráfaga de CPU. Es posible realizar una estimación de la próxima, utilizando la media de las ráfagas de CPU para cada proceso. Así, por ejemplo, podemos tener la siguiente fórmula (1):

$$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

Donde:

T_i = duración de la ráfaga de CPU i-ésima del proceso.

S_i = valor estimado para el i-ésimo caso

S_i = valor estimado para la primera ráfaga de CPU. No es calculado.

    a. Suponga un proceso cuyas ráfagas de CPU reales tienen como duración: 6, 4, 6, 4, 13, 13, 13. Calcule qué valores se obtendrían como estimación para las ráfagas de CPU del proceso si se utiliza la fórmula 1, con un valor inicial

estimado de S_1=10.

La fórmula 1 le da el mismo peso a todos los casos (siempre calcula la media). Es posible reescribirla permitiendo darle un peso mayor a los casos más recientes y menor a casos viejos (o viceversa). Se plantea la siguiente fórmula 2:

$$S_{n+1} = \alpha T_n + (1 - \alpha) S_n$$

Con $0 < \alpha < 1$.

b. Analice para qué valores de α se tienen en cuenta los casos más recientes.
c. Para la situación planteada en a) calcule qué valores se obtendrían si se utiliza la fórmula 2 con $\alpha = 0,2$; $\alpha = 0,5$ y $\alpha = 0,8$.
d. Para todas las estimaciones realizadas en a y c ¿Cuál es la que más se asemeja a las ráfagas de CPU reales del proceso?

13. Colas Multinivel

Actualmente los algoritmos de planificación vistos se han ido combinando para formar algoritmos más eficientes. Así surge el algoritmo de Colas Multinivel, donde la cola de procesos listos es dividida en varias colas, teniendo cada una su propio algoritmo de planificación.

a. Suponga que se tiene dos tipos de procesos: *Interactivos* y *Batch*. Cada uno de estos procesos se coloca en una cola según su tipo. ¿Qué algoritmo de los vistos utilizará para administrar cada una de estas colas?.

A su vez, se utiliza un algoritmo para administrar cada cola que se crea. Así, por ejemplo, el algoritmo podría determinar mediante prioridades sobre qué cola elegir un proceso.

b. Para el caso de las dos colas vistas en a: ¿Qué algoritmo utilizaría para planificarlas?

14. Suponga que en un Kernel se utiliza un algoritmo de planificación de colas multinivel. El mismo cuenta con 3 colas de procesos listos, en las que los procesos se encolan en una u otra según su prioridad. Hay 3 prioridades (1 , 2 , 3), donde un menor número indica mayor prioridad. Se utiliza el algoritmo de prioridades para la administración entre las colas.

Se tiene el siguiente lote de procesos a ser procesados con sus respectivas operaciones de I/O:

Job	Llegada	CPU	I/O (rec,ins,dur)	Prioridad
1	0	9	(R1, 4, 2) (R2, 6, 3) (R1, 8, 3)	1
2	1	5	(R3, 3, 2) (R3, 4, 2)	2
3	2	5	(R1, 4, 1)	3
4	3	7	(R2, 1, 2) (R2, 5, 3)	2
5	5	5	(R1, 2, 3) (R3, 4, 3)	1

Suponiendo que las colas de cada recurso (dispositivo de entrada/salida) se

administran a través de FCFS y que cada cola de procesos listos se administra por medio de un algoritmo RR con un quantum de 3 unidades, realice un diagrama de Gantt:

a. Asumiendo que NO hay apropiación entre los procesos.
b. Asumiendo que hay apropiación entre los procesos.

15. En el esquema de Colas Multinivel, cuando se utiliza un algoritmo de prioridades para administrar las diferentes colas los procesos pueden sufrir starvation.

La técnica de envejecimiento se puede aplicar a este esquema, haciendo que un proceso cambie de una cola de menor prioridad a una de mayor prioridad, después de cierto periodo de tiempo que el mismo se encuentra esperando en su cola. Luego de llegar a una cola en la que el proceso llega a ser atendido, el mismo retorna a su cola original.

Por ejemplo: Un proceso con prioridad 3 está en cola su cola correspondiente. Luego de X unidades de tiempo, el proceso se mueve a la cola de prioridad 2. Si en esta cola es atendido, retorna a su cola original, en caso contrario luego de sucederse otras X unidades de tiempo el proceso se mueve a la cola de prioridad 1. Esta última acción se repite hasta que el proceso obtiene la CPU, situación que hace que el mismo vuelva a su cola original.

Para los casos a y b del ejercicio 15 realice el diagrama de Gantt considerando además que se tiene un envejecimiento de 4 unidades.

16. La situación planteada en el ejercicio 16, donde un proceso puede cambiar de una cola a otra, se la conoce como Colas Multinivel con Realimentación.

Suponga que se quiere implementar un algoritmo de planificación que tenga en cuenta el tiempo de ejecución consumido por el proceso, penalizando a los que más tiempo de ejecución tienen. (Similar a la tarea del algoritmo SJF que tiene en cuenta el tiempo de ejecución que resta).

Utilizando los conceptos vistos de Colas Multinivel con Realimentación indique qué colas implementaría, qué algoritmo usaría para cada una de ellas así como para la administración de las colas entre sí.

Tenga en cuenta que los procesos no deben sufrir inanición.

17. A cuáles de los siguientes tipos de trabajos:

- cortos acotados por CPU
- cortos acotados por E/S
- largos acotados por CPU
- largos acotados por E/S

benefician las siguientes estrategias de administración:

a. prioridad determinada estáticamente con el método del más corto primero (SJF).
b. prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.

18. Analizar y explicar por qué si el quantum en Round-Robin se incrementa sin límite, el algoritmo de planificación se aproxima a FIFO.

19. Dado los siguientes programas en un pseudo-código que simula la utilización de llamadas al sistema para crear procesos en Unix/Linux, indicar qué mensajes se imprimirán en pantalla (sin importar el orden en que saldrían):
    a. Caso 1

```
printf("hola")
x = fork()
if x <  1 {
      execv("ls")
      printf("mundo")
      exit(0)
}
exit(0)
```

    b. Caso 2

```
printf("hola")
x = fork()
if x <  1 {
      execv("ps")
       printf("mundo")
       exit(0)
}
execv("ls")
printf("fin")
exit(0)
```

    c. Caso 3

```
printf("Anda    a   rendir   el   Primer   Parcial   de
Promo!")
newpid = fork()
if newpid == 0 {
    printf("Estoy comenzando el Examen")
    execv("ps")
    printf("Termine el Examen")
}
printf("¿Como te fue?")
exit(0)
printf("Ahora anda a descansar")
```

20. Dado el siguiente programa en C, analice su objetivo sin necesidad de ejecutarlo. (Investigue el funcionamiento del iterador *for* en C para poder responder: https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap4/cap43.htm)

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main ( void ) {
     int c ;
     pid_t pid ;
     printf( " Comienzo . : \ n " ) ;
     for ( c = 0 ; c < 3 ; c++ ) {
```

```
            pid = fork ( ) ;
        }
        printf ( " Proceso \n " ) ;
        return 0 ;
}
```

a. ¿Cuántas líneas con la palabra "Proceso" aparecen al final de la ejecución de este programa?
b. *¿El número de líneas es el número de procesos que han estado en ejecución?.* Ejecute el programa y compruebe si su respuesta es correcta. Modifique el valor del bucle for y compruebe los nuevos resultados.

21. Modifiquemos el programa anterior. Ahora, además de un mensaje, vamos a añadir una variable y antes de finalizar el programa vamos a mostrar el valor de la misma:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main ( void ) {
      int c ;
      pid_t pid ;
      int p = 0;
      printf( " Comienzo . : \ n " ) ;
      for ( c = 0 ; c < 3 ; c++ ) {
            pid = fork ( ) ;
      }
      p++;
      printf ( " Proceso %d\n ",p ) ;
      return 0 ;
}
```

a. ¿Qué valores se imprimirán en la consola?
b. ¿Todas las líneas tendrán el mismo valor o algunas líneas tendrán valores distintos?
c. ¿Cuál es el valor (o valores) que aparece*?.* Compile y ejecute el programa y compruebe si su respuesta es correcta.
d. Realice modificaciones al programa, por ejemplo: modifique el valor del bucle for, cambie el lugar dónde se incrementa la variable p, y compruebe los nuevos resultados.

22. ¿Qué es la MMU y qué funciones cumple?

23. ¿Que representa el espacio de direcciones de un proceso?

24. Explique y relacione los siguientes conceptos:
    Dirección Lógica o Virtual
    Dirección Física

25. En la técnica de Particiones Múltiples, la memoria es dividida en varias particiones y

los espacios de direcciones de los procesos son ubicados en estas, siempre que el tamaño del mismo sea menor o igual que el tamaño de la partición.

Al trabajar con particiones se pueden considerar 2 métodos (independientes entre sí):

Particiones Fijas
Particiones Dinámicas

a. Explique cómo trabajan estos 2 métodos. Cite diferencias, ventajas y desventajas.
b. ¿Qué información debe disponer el Kernel para poder administrar la memoria principal con estos métodos?
c. Realice un gráfico indicando cómo se realiza la transformación de direcciones lógicas a direcciones físicas.

26. Al trabajar con particiones fijas, los tamaños de las mismas se pueden considerar:
    - ➢ Particiones de igual tamaño.
    - ➢ Particiones de diferente tamaño.

Cite ventajas y desventajas entre las alternativas.

27. Fragmentación. Ambos métodos de particiones presentan el problema de la fragmentación:

Fragmentación Interna (Para el caso de Particiones Fijas)
Fragmentación Externa (Para el caso de Particiones Dinámicas)

a. Explique a qué hacen referencia estos 2 problemas
b. El problema de la Fragmentación Externa es posible de subsanar. Explique una posible técnica que permita al Kernel mitigar este problema.

28. Analice y describa cómo funcionan las siguientes técnicas de asignación de particiones: Best Fit, Worst Fit, First Fit y Next Fit. Tenga en cuenta que información debe mantener el Kernel. Indique, para los métodos de particiones dinámicas y fijas, con cuál técnica se obtienen mejores resultados y justifique.

29. Segmentación
    a. Explique cómo trabaja este método de asignación de memoria.
    b. ¿Qué estructuras son necesarias en el Kernel para poder ejecutarse sobre un Hardware que utiliza segmentación?
    c. Explique, utilizando gráficos, cómo son resueltas las direcciones lógicas a físicas.
    d. En este esquema: ¿se puede producir fragmentación (interna y/o externa)?
    e. De las técnicas enumeradas en el ejercicio 29 ¿Cuál se ajusta mejor para la asignación de memoria principal?

30. Dado un esquema de segmentación donde cada dirección hace referencia a 1 byte y la siguiente tabla de segmentos de un proceso, traduzca, de corresponder, las

direcciones lógicas indicadas a direcciones físicas. La Dirección lógica está representada por: **segmento:desplazamiento**

Segmento	Dir. Base	Tamaño
0	102	12500
1	28699	24300
2	68010	15855
3	80001	400

i. 0000:9001
ii. 0001:24301
iii. 0002:5678
iv. 0001:18976
v. 0003:0

31. Paginación
    a. Explique cómo trabaja este método de asignación de memoria.
    b. ¿Qué estructuras son necesarias en el Kernel para poder ejecutarse sobre un Hardware que utiliza paginación ?
    c. Explique, utilizando gráficos, cómo son resueltas las direcciones lógicas en físicas.
    d. En este esquema: ¿se puede producir fragmentación (interna y/o externa)?

32. Suponga un sistema donde la memoria es administrada mediante la técnica de paginación, y donde:
    El tamaño de la página es de 512 bytes
    Cada dirección de memoria referencia 1 byte.
    Se tiene una memoria principal de 10240 bytes (10 KiB) y los marcos se enumeran desde 0 y comienzan en la dirección física 0.

Suponga además un proceso P1 con un tamaño lógico de 2000 bytes y con la siguiente tabla de páginas:

Página	Marco
0	3
1	5
2	2
3	6

    a. Realice los gráficos necesarios (de la memoria principal, del espacio de direcciones del proceso, de tabla de páginas y de la tabla de marcos) en el que refleje el estado descripto.
    b. Indicar si las siguientes direcciones lógicas corresponden al espacio lógico del proceso P1 y en caso afirmativo indicar la dirección física a la que corresponden:
        i. 35
        ii. 512
        iii. 2051
        iv. 0
        v. 1325
        vi. 602

c. Indicar, en caso de ser posible, las direcciones lógicas del proceso P1 que se corresponden a las siguientes direcciones físicas:
   i. 509
   ii. 1500
   iii. 0
   iv. 3215
   v. 1024
   vi. 2000

33. Dado un esquema donde cada dirección hace referencia a 1 byte, con páginas de 2 KiB (KibiBytes), donde el frame 0 se encuentra en la dirección física 0. Con las siguientes siguientes primeras entradas de la tabla de páginas de un proceso,

Página	Marco
0	16
1	13
2	9
3	2
4	0

traduzca las direcciones lógicas indicadas a direcciones físicas:
   i. 5120
   ii. 3242
   iii. 1578
   iv. 2048
   v. 8191

34. Tamaño de la Página. Compare las siguientes situaciones con respecto al tamaño de página y su impacto, indicando ventajas y desventajas:
   a. Un tamaño de página pequeño.
   b. Un tamaño de página grande.

35. Existen varios enfoques para administrar las tablas de páginas:
   - Tablas de páginas de 1 nivel.
   - Tablas de páginas de 2 niveles o más
   - Tablas de páginas invertidas.

Explique brevemente cómo trabajan estos mecanismos. Tenga en cuenta analizar cómo se resuelven las direcciones y qué ventajas/desventajas presentan una sobre otra.

36. Dado un esquema de paginación, donde cada dirección hace referencia a 1 byte, se tiene un tamaño de dirección de 32 bits y un tamaño de página de 2048 bytes. Suponga además un proceso P1 que necesita 51358 bytes para su código y 68131 bytes para sus datos.
   a. De los 32 bits de las direcciones ¿Cuántos se utilizan para identificar página? ¿Cuantos para desplazamiento?
   b. ¿Cuál sería el tamaño lógico máximo de un proceso?
   c. ¿Cuántas páginas máximo puede tener el proceso P1?
   d. Si se contaran con 4GiB (Gibibytes) de RAM, ¿cuantos marcos habría disponibles para utilizar?
   e. ¿Cuántas páginas necesita P1 para almacenar su código?
   f. ¿Cuántas páginas necesita P1 para almacenar sus datos?

g. ¿Cuántos bytes habría de fragmentación interna entre lo necesario para almacenar su código y sus datos? Ayuda: Recuerde que en una misma página no pueden convivir código y datos.
h. Asumiendo que el hardware utiliza tabla de páginas de un (1) nivel, que la dirección se interpreta como 20 bits para identificar página y 12 bits para desplazamiento, y que cada Entrada de Tabla de Páginas (PTE) requiere de 1 Kib. ¿Cuál será el tamaño mínimo que la tabla de páginas del proceso P1 ocupará en RAM?
i. Asumiendo que el hardware utiliza tabla de páginas de dos (2) niveles, que la dirección se interpreta como 10 bits para identificar primer nivel, 10 bits para identificar el segundo nivel, 12 bits para desplazamiento y que cada Entrada de Tabla de Páginas (PTE) requiere de 1 Kib. ¿Cuál será el tamaño mínimo que la tabla de páginas del proceso P1 ocupará en RAM?
j. Dado los resultados obtenidos en los dos incisos anteriores (h, i) ¿Qué conclusiones se pueden obtener del uso de los 2 modelos de tablas de página?

37. Cite similitudes y diferencias entre las 4 técnicas vistas: Particiones Fijas, Particiones Dinámicas, Segmentación y Paginación.

38. Dada la técnica de administración de memoria por medio de segmentación paginada:
    a. Explique cómo funciona.
    b. Cite ventajas y desventajas respecto a las técnicas antes vistas.
    c. Teniéndose disponibles las siguientes tablas:

Tabla de Segmentos

Núm. Seg.	Dir. base
1	500
2	1500
3	5000

Tabla de Páginas

Nro. Segmento	Nro. Página	Direc. Base
1	1	40
	2	80
	3	60
2	1	20
	2	25
	3	0
3	1	120
	2	150

Indicar las direcciones físicas correspondientes a las siguientes direcciones lógicas (segmento,página,desplazamiento):

i. (2,1,1)
ii. (1,3,15)
iii. (3,1,10)
iv. (2,3,5)

Conceptos de Sistemas Operativos (IC)

Introducción a los Sistemas Operativos (LI, LS, APU, ATIC)

Trabajo Práctico N° 3

Objetivo

Introducir a los estudiantes en el armado y ejecución de scripts en bash, Shell Scripting.

Temas incluidos

Bash. Script. Sintaxis. GNU/Linux

1. ¿Qué es el Shell Scripting? ¿A qué tipos de tareas están orientados los script? ¿Los scripts deben compilarse? ¿Por qué?
2. Investigar la funcionalidad de los comandos echo y read.
   a. ¿Cómo se indican los comentarios dentro de un script?
   b. ¿Cómo se declaran y se hace referencia a variables dentro de un script?
3. Crear dentro del directorio personal del usuario logueado un directorio llamado *practica-shell-script* y dentro de él un archivo llamado mostrar.sh cuyo contenido sea el siguiente:

```
#!/bin/bash
# Comentarios acerca de lo que hace el script
# Siempre comento mis scripts, si no lo hago hoy,
# mañana ya no me acuerdo de lo que quise hacer
echo "Introduzca su nombre y apellido:"
read nombre apellido
echo "Fecha y hora actual:"
date
echo "Su apellido y nombre es:
echo "$apellido $nombre"
echo "Su usuario es: `whoami`"
echo "Su directorio actual es:"
```

   a. Asignar al archivo creado los permisos necesarios de manera que pueda ejecutarlo
   b. Ejecutar el archivo creado de la siguiente manera: ./mostrar
   c. ¿Qué resultado visualiza?
   d. Las backquotes (`) entre el comando whoami ilustran el uso de la sustitución de comandos. ¿Qué significa esto?
   e. Realizar modificaciones al script anteriormente creado de manera de poder mostrar distintos resultados (cuál es su directorio personal, el contenido de un directorio en particular, el espacio libre en disco, etc.). Pida que se introduzcan por teclado (entrada estándar) otros datos.

4. Parametrización: ¿Cómo se acceden a los parámetros enviados al script al momento de su invocación? ¿Qué información contienen las variables $#, $*, $? y $HOME dentro de un script?

5. ¿Cual es la funcionalidad de comando exit? ¿Qué valores recibe como parámetro y cuál es su significado?

6. El comando *expr* permite la evaluación de expresiones. Su sintaxis es:
   expr arg1 op arg2,
   donde arg1 y arg2 representan argumentos y op la operación de la expresión. Investigar que tipo de operaciones se pueden utilizar.

7. El comando *`test expresion`* permite evaluar expresiones y generar un valor de retorno, true o false. Este comando puede ser reemplazado por el uso de corchetes de la siguiente manera [expresión]. Investigar qué tipo de expresiones pueden ser usadas con el comando test. Tenga en cuenta operaciones para: evaluación de archivos, evaluación de cadenas de caracteres y evaluaciones numéricas.

8. Estructuras de control. Investigue la sintaxis de las siguientes estructuras de control incluidas en shell scripting:
   - ➢ if
   - ➢ case
   - ➢ while
   - ➢ for
   - ➢ select
9. ¿Qué acciones realizan las sentencias *break* y *continue* dentro de un bucle? ¿Qué parámetros reciben?

10. ¿Qué tipo de variables existen? ¿Es shell script fuertemente tipado? ¿Se pueden definir arreglos? ¿Cómo?

11. ¿Pueden definirse funciones dentro de un script? ¿Cómo? ¿Cómo se maneja el pasaje de parámetros de una función a la otra?

12. Evaluación de expresiones.
    a. Realizar un script que le solicite al usuario 2 números, los lea de la entrada Standard e imprima la multiplicación, suma, resta y cual es el mayor de los números leídos.
    b. Modificar el script creado en el inciso anterior para que los números sean

recibidos como parámetros. El script debe controlar que los dos parámetros sean enviados.

c. Realizar una calculadora que ejecute las 4 operaciones básicas: +, - ,*, %. Esta calculadora debe funcionar recibiendo la operación y los números como parámetros

13. Uso de las estructuras de control:
    a. Realizar un script que visualice por pantalla los números del 1 al 100 así como sus cuadrados.
    b. Crear un script que muestre 3 opciones al usuario: Listar, DondeEstoy y QuienEsta. Según la opción elegida se le debe mostrar:
        - ➢ Listar: lista el contenido del directorio actual.
        - ➢ DondeEstoy: muestra la ruta deldirectorio donde me encuentro ubicado.
        - ➢ QuienEsta: muestra los usuarios conectados al sistema.
    c. Crear un script que reciba como parámetro el nombre de un archivo e informe si el mismo existe o no, y en caso afirmativo indique si es un directorio o un archivo. En caso de que no exista el archivo/directorio cree un directorio con el nombre recibido como parámetro.

14. Renombrando Archivos: haga un script que renombre solo archivos de un directorio pasado como parámetro, agregandole una CADENA, contemplando las opciones:
    - ➢ "-a CADENA": renombra el fichero concatenando CADENA al final del nombre del archivo
    - ➢ "-b CADENA": renombra el fichero concatenando CADENA al comienzo del nombre del archivo

    Ejemplos:

    Si tengo los siguientes archivos: /tmp/a /tmp/b , al ejecutar: **./renombra /tmp/ -a EJ** obtendré como resultado: /tmp/aEJ /tmp/bEJ. Y si ejecuto: **./renombra /tmp/ -b EJ** el resultado será: /tmp/EJa /tmp/EJb

15. El comando *cut* nos permite procesar las líneas de la entrada que reciba (archivo, entrada estándar, resultado de otro comando, etc) y cortar columnas o campos, siendo posible indicar cuál es el delimitador de las mismas. Investigue los parámetros que puede recibir este comando y cite ejemplos de uso.

16. Realizar un script que reciba como parámetro una extensión y haga un reporte con 2 columnas, el nombre de usuario y la cantidad de archivos que posee con esa extensión. Se debe guardar el resultado en un archivo llamado *reporte.txt*

17. Escribir un script que al ejecutarse imprima en pantalla los nombre de los

archivos que se encuentran en el directorio actual, intercambiando minúsculas por mayúsculas, además de eliminar la letra a (mayúscula o minúscula).

Por ejemplo, si en el directorio actual están los siguientes archivos:

- IsO
- pepE
- Maria

y ejecutó: ./ejercicio17 , se obtendrá como resultado:

- iSo
- PEPe
- mRI

Ayuda: Investigar el comando *tr*

18. Crear un script que verifique cada 10 segundos si un usuario se ha logueado en el sistema (el nombre del usuario será pasado por parámetro). Cuando el usuario finalmente se loguee, el programa deberá mostrar el mensaje "Usuario XXX logueado en el sistema" y salir.

19. Escribir un Programa de "Menu de Comandos Amigable con el Usuario" llamado menú, el cual, al ser invocado, mostrará un menú con la selección para cada uno de los scripts creados en esta práctica. Las instrucciones de cómo proceder deben mostrarse junto con el menú. El menú deberá iniciarse y permanecer activo hasta que se seleccione Salir. Por ejemplo:

```
MENU DE COMANDOS
03. Ejercicio 3
12. Evaluar Expresiones
13. Probar estructuras de control
...
Ingrese la opción a ejecutar: 03
```

20. Realice un script que simule el comportamiento de una estructura de PILA e implemente las siguientes funciones aplicables sobre una estructura global definida en el script:
    - ➔ push: Recibe un parámetro y lo agrega en la pila
    - ➔ pop: Saca un elemento de la pila
    - ➔ length: Devuelve la longitud de la pila
    - ➔ print: Imprime todos elementos de la pila

    Dentro del mismo script y utilizando las funciones implementadas:

    1. Agregue 10 elementos a la pila

2. Saque 3 de ellos
3. Imprima la longitud de la pila
4. Luego imprima la totalidad de los elementos que en ella se encuentran.

21. Dada la siguiente declaración al comienzo de un script:

```
num=(10 3 5 7 9 3 5 4)
```

(la cantidad de elementos del arreglo puede variar).

Implemente la función **productoria** dentro de este script, cuya tarea sea multiplicar todos los números que el arreglo contiene.

22. Implemente un script que recorra un arreglo compuesto por números e imprima en pantalla sólo los números pares y que cuente sólo los números impares y los informe en pantalla al finalizar el recorrido.

23. Dada la definición de 2 vectores del mismo tamaño y cuyas longitudes no se conocen.
    vector1=(1 .. N)
    vector2=(1.. N)

Por ejemplo:
    vector1=(1 80 65 35 2) y vector2=(5 98 3 41 8).

Complete este script de manera tal de implementar la suma elemento a elemento entre ambos vectores y que la misma sea impresa en pantalla de la siguiente manera:

- ➢ La suma de los elementos de la posición 0 de los vectores es 6
- ➢ La suma de los elementos de la posición 1 de los vectores es 178 ...
- ➢ La suma de los elementos de la posición 4 de los vectores es 10

24. Realice un script que agregue en un arreglo todos los nombres de los usuarios del sistema pertenecientes al grupo “users”. Adicionalmente el script puede recibir como parámetro:
    - ➢ “-b n”: Retorna el elemento de la posición n del arreglo si el mismo existe. Caso contrario, un mensaje de error.
    - ➢ “-l”: Devuelve la longitud del arreglo
    - ➢ “-i”: Imprime todos los elementos del arreglo en pantalla

25. Escriba un script que reciba una cantidad desconocida de parámetros al momento de su invocación (debe validar que al menos se reciba uno). Cada parámetro representa la ruta absoluta de un archivo o directorio en el sistema. El script deberá iterar por todos los parámetros recibidos, y solo para aquellos parámetros que se encuentren en posiciones impares (el primero, el tercero, el

verificar si el archivo o directorio existen en el sistema, imprimiendo en pantalla que tipo de objeto es (archivo o directorio). Además, deberá informar la cantidad de archivos o directorios inexistentes en el sistema.

26. Realice un script que implemente a través de la utilización de funciones las operaciones básicas sobre arreglos:
    - ➢ inicializar: Crea un arreglo llamado array vacío
    - ➢ agregar_elem <parametro1>: Agrega al final del arreglo el parámetro recibido
    - ➢ eliminar_elem <parametro1>: Elimina del arreglo el elemento que se encuentra en la posición recibida como parámetro. Debe validar que se reciba una posición válida
    - ➢ longitud: Imprime la longitud del arreglo en pantalla
    - ➢ imprimir: Imprime todos los elementos del arreglo en pantalla
    - ➢ inicializar_Con_Valores <parametro1><parametro2>: Crea un arreglo con longitud <parametro1>y en todas las posiciones asigna el valor <parametro2>

27. Realice un script que reciba como parámetro el nombre de un directorio. Deberá validar que el mismo exista y de no existir causar la terminación del script con código de error 4. Si el directorio existe deberá contar por separado la cantidad de archivos que en él se encuentran para los cuales el usuario que ejecuta el script tiene permiso de lectura y escritura, e informar dichos valores en pantalla. En caso de encontrar subdirectorios, no deberán procesarse, y tampoco deberán ser tenidos en cuenta para la suma a informar.

28. Implemente un script que agregue a un arreglo todos los archivos del directorio /home cuya terminación sea .doc. Adicionalmente, implemente las siguientes funciones que le permitan acceder a la estructura creada:
    - ➢ verArchivo <nombre_de_archivo>: Imprime el archivo en pantalla si el mismo se encuentra en el arreglo. Caso contrario imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 5
    - ➢ cantidadArchivos: Imprime la cantidad de archivos del /home con terminación .doc
    - ➢ borrarArchivo <nombre_de_archivo>: Consulta al usuario si quiere eliminar el archivo lógicamente. Si el usuario responde Si, elimina el elemento solo del arreglo. Si el usuario responde No, elimina el archivo del arreglo y también del FileSystem. Debe validar que el archivo exista en el arreglo. En caso de no existir, imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 10

29. Realice un script que mueva todos los programas del directorio actual (archivos

ejecutables) hacia el subdirectorio “bin” del directorio HOME del usuario actualmente logueado. El script debe imprimir en pantalla los nombres de los que mueve, e indicar cuántos ha movido, o que no ha movido ninguno. Si el directorio “bin” no existe,deberá ser creado.

30. Implemente la estructura de datos Set (Conjunto de valores) en Bash. Un conjunto se define como una colección de valores únicos, es decir que solo almacena una vez cada valor, aún cuando se intente agregar el mismo valor más de una vez. La implementación debe soportar las siguientes operaciones mediante funciones:
    - **initialize** - inicializa el set vacío.
    - **initialize_with** - inicializa el set con un conjunto de valores que recibe como argumento (debe validar que se reciba al menos uno).
    - **add** - Agrega un valor al conjunto, el cual recibe como argumento. No debe agregar elementos repetidos. El resultado de la operación será un éxito solo si el valor puede ser agregado al conjunto.
    - **remove** - Elimina uno o más valores del conjunto, los cuales recibe como argumentos. Si la operación elimina al menos un valor, se considera un éxito.
    - **contains** - Chequea si el conjunto contiene un valor recibido como argumento. El resultado será éxito si el valor está en el conjunto.
    - **print** - Imprime los elementos del conjunto, de a uno por línea.
    - **print_sorted** - Imprime los elementos del conjunto, de a uno por línea y ordenados alfabéticamente. *Tip: Investigar cómo combinar el comando sort con la función print.*

    En un script separado, incorporar y utilizar las funciones implementadas para desarrollar un juego de bingo. El bingo deberá generar números aleatorios dentro de un rango entre 0 y un valor máximo que puede especificarse mediante un argumento del script, de manera opcional. El valor máximo no puede ser 0 ni superior a 32767, y en caso de no especificarse se tomará como valor por defecto 99. En cada ronda se generará un nuevo número que ya no haya sido utilizado y se lo cantará, imprimiendo en la salida estándar. Luego de esto, se esperará entrada del usuario para saber si se debe cantar “BINGO” para finalizar la partida o se debe cantar un nuevo numero. Al finalizar, el script debera imprimir en orden los números se cantaron hasta que se produjo el bingo.
    Tip: Investigar la variable de entorno $RANDOM de Bash para obtener valores aleatorios.

31. Realice un script que reciba como argumento una lista de posibles nombres de usuarios del sistema y, para cada uno de los que efectivamente existan en el sistema y posean un directorio personal configurado que sea válido, realice las modificaciones necesarias en su directorio personal para que tenga un subdirectorio llamado “directorio_iso” con la siguiente estructura:

```
directorio_iso
├── 2025
│   ├── 01
│   │   └── archivo.txt
│   ├── 02
│   │   └── archivo.txt
│   ├── 03
│   │   └── archivo.txt
│   ├── 04
│   │   └── archivo.txt
│   ├── 05
│   │   └── archivo.txt
│   ├── 06
│   │   └── archivo.txt
│   ├── 07
│   │   └── archivo.txt
│   ├── 08
│   │   └── archivo.txt
│   ├── 09
│   │   └── archivo.txt
│   ├── 10
│   │   └── archivo.txt
│   ├── 11
│   │   └── archivo.txt
│   └── 12
│       └── archivo.txt
└── 2026
    ├── 01
    │   └── archivo.txt
    ├── 02
    │   └── archivo.txt
    ├── 03
    │   └── archivo.txt
    ├── 04
    │   └── archivo.txt
    ├── 05
    │   └── archivo.txt
    ├── 06
    │   └── archivo.txt
    ├── 07
    │   └── archivo.txt
    ├── 08
    │   └── archivo.txt
    ├── 09
```

```
    │   └── archivo.txt
    ├── 10
    │   └── archivo.txt
    ├── 11
    │   └── archivo.txt
    └── 12
        └── archivo.txt
```

Para resolver la creación de los directorios y archivos utilice la funcionalidad “Brace Expansion” brindada por bash:
https://www.gnu.org/software/bash/manual/html_node/Brace-Expansion.html

Conceptos de Sistemas Operativos (IC)

Introducción a los Sistemas Operativos (LI, LS, APU, ATIC)

Trabajo Práctico N° 1

Objetivo

El objetivo de esta práctica es que el estudiante se familiarice con los conceptos básicos del sistema operativo *GNU/Linux*, su instalación, entorno y comandos principales. Manejo de Usuarios, permisos y su sistemas de archivos.

Temas Incluidos

GNU/Linux, instalación y conceptos básicos, permisos, arranque, usuarios. organización interna.

1. Características de *GNU/Linux*:
   a. Mencione y explique las características más relevantes de *GNU/Linux*.
   b. Mencione otros sistemas operativos y compárelos con *GNU/Linux* en cuanto a los puntos mencionados en el inciso *a*.
   c. ¿Qué es **GNU**?
   d. Indique una breve historia sobre la evolución del proyecto *GNU.*
   e. Explique qué es la multitarea, e indique si *GNU/Linux* hace uso de ella.
   f. ¿Qué es **POSIX**?

2. Distribuciones de *GNU/Linux*:
   a. ¿Qué es una distribución de *GNU/Linux*? Nombre al menos 4 distribuciones de *GNU/Linux* y cite diferencias básicas entre ellas.
   b. ¿En qué se diferencia una distribución de otra?
   c. ¿Qué es **Debian**? Acceda al sitio https://www.debian.org/ e indique cuáles son los objetivos del proyecto y una breve cronología del mismo.

3. Estructura de *GNU/Linux*:
   a. Nombre cuáles son los 3 componentes fundamentales de *GNU/Linux*.
   b. Mencione y explique la estructura básica del Sistema Operativo *GNU/Linux*.

4. Kernel:
   a. ¿Cuáles son sus funciones principales?
   b. ¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado del Kernel en versiones anteriores a la 2.4? ¿Qué cambió en el versionado que se impuso a partir de la versión 2.6?
   c. ¿Es posible tener más de un Kernel de *GNU/Linux* instalado en la misma máquina?
   d. ¿Dónde se encuentra ubicado dentro del File System?

5. Intérprete de comandos *(Shell)*:
    a. ¿Qué es?
    b. ¿Cuáles son sus funciones?
    c. Mencione al menos 3 intérpretes de comandos que posee *GNU/Linux* y compárelos entre ellos.
    d. ¿Dónde se ubican (*path*) los comandos propios y externos al Shell?
    e. ¿Por qué considera que el Shell no es parte del Kernel de *GNU/Linux*?
    f. ¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde se define? ¿Cualquier usuario puede realizar dicha tarea?

6. El sistema de Archivos *(File System)* en Linux:
    a. ¿Qué es?
    b. ¿Cuál es la estructura básica de los File System en *GNU/Linux*? Mencione los directorios más importantes e indique qué tipo de información se encuentra en ellos. ¿A qué hace referencia la sigla **FHS**?
    c. Mencione sistemas de archivos soportados por GNU/Linux.
    d. ¿Es posible visualizar particiones del tipo FAT y NTFS (que son de Windows) en GNU/Linux?

7. Particiones:
    a. Definición. Tipos de particiones. Ventajas y Desventajas.
    b. ¿Cómo se identifican las particiones en *GNU/Linux*? (Considere discos **IDE**, **SCSI** y **SATA**).
    c. ¿Cuántas particiones son necesarias como mínimo para instalar *GNU/Linux*? Nómbrelas indicando tipo de partición, identificación, tipo de File System y punto de montaje.
    d. Dar ejemplos de diversos casos de particionamiento dependiendo del tipo de tarea que se deba realizar en su sistema operativo.
    e. ¿Qué tipo de software para particionar existe? Menciónelos y compare.

8. Arranque (*bootstrap*) de un Sistema Operativo:
    a. ¿Qué es el **BIOS**? ¿Qué tarea realiza?
    b. ¿Qué es **UEFI**? ¿Cuál es su función?
    c. ¿Qué es el **MBR**? ¿Qué es el **MBC**?
    d. ¿A qué hacen referencia las siglas **GPT**? ¿Qué sustituye? Indique cuál es su formato.
    e. ¿Cuál es la funcionalidad de un "Gestor de Arranque"? ¿Qué tipos existen? ¿Dónde se instalan? Cite gestores de arranque conocidos.
    f. ¿Cuáles son los pasos que se suceden desde que se prende una computadora hasta que el Sistema Operativo es cargado (proceso de *bootstrap*)?
    g. Analice el proceso de arranque en *GNU/Linux* y describa sus principales pasos.
    h. ¿Cuáles son los pasos que se suceden en el proceso de parada (*shutdown*) de

GNU/Linux?

i. ¿Es posible tener en una PC *GNU/Linux* y otro Sistema Operativo instalado? Justifique.

9. Archivos y editores:
   a. ¿Cómo se identifican los archivos en *GNU/Linux*?
   b. Investigue el funcionamiento de los editores **vim, nano** y **mcedit**, y los comandos **cat, more y less**.
   c. Cree un archivo llamado "prueba.exe" en su directorio personal usando el **vim**. El mismo debe contener su número de alumno y su nombre.
   d. Investigue el funcionamiento del comando **file**. Pruébelo con diferentes archivos. ¿Qué diferencia nota?
   e. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso de archivos:
      i. cd
      ii. mkdir
      iii. rmdir
      iv. ln
      v. tail
      vi. locate
      vii. ls
      viii. pwd
      ix. cp
      x. mv
      xi. find

10. Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones. Investigue su funcionamiento y parámetros más importantes:
    a. Cree la carpeta **ISOCSO**
    b. Acceda a la carpeta
    c. Cree dos archivos con los nombres **isocso.txt** e **isocso.csv**
    d. Liste el contenido del directorio actual
    e. Visualizar la ruta donde estoy situado
    f. Busque todos los archivos en los que su nombre contiene la cadena "iso*"
    g. Informar la cantidad de espacio libre en disco
    h. Verifique los usuarios conectados al sistema
    i. Editar a el archivo **isocso.txt** e ingresar Nombre y Apellido
    j. Mostrar en pantalla las últimas líneas de un archivo.

11. Investigue el funcionamiento, parámetros y ubicación (directorio) de los siguientes comandos:

- ➔ man
- ➔ shutdown
- ➔ reboot
- ➔ halt
- ➔ uname
- ➔ dmesg
- ➔ lspci
- ➔ at
- ➔ netstat
- ➔ head
- ➔ tail

12. Procesos:
    a. ¿Qué es un proceso? ¿A que hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.
    b. Investigue el funcionamiento, parámetros y ubicación (directorio) de los siguientes comandos relacionados a procesos. En caso de que algún comando no venga por defecto en la distribución que utiliza deberá proceder a instalarlo:

        i. top
        ii. htop
        iii. ps
        iv. pstree
        v. kill
        vi. pgrep
        vii. pkill
        viii. killall
        ix. renice
        x. xkill
        xi. atop
        xii. nice

13. Proceso de Arranque *SystemV* (https://github.com/systeminit/si):
    a. Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.
    b. Proceso **INIT**. ¿Quién lo ejecuta? ¿Cuál es su objetivo?
    c. RunLevels. ¿Qué son? ¿Cuál es su objetivo?
    d. ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?
    e. Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en él se almacena?
    f. Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) deberá ejecutar para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?
    g. Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.

14. *SystemD* (https://github.com/systemd/systemd):
    a. ¿Qué es *systemd*?
    b. ¿A qué hace referencia el concepto de *Unit* en SystemD?
    c. ¿Para qué sirve el comando *systemctl* en SystemD?
    d. ¿A qué hace referencia el concepto de *target* en SystemD?
    e. Ejecutar el comando *pstree*. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?

15. Usuarios:
    a. ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?
    b. ¿A qué hacen referencia las siglas *UID* y *GID*? ¿Pueden coexistir UIDs iguales en un

sistema GNU/Linux? Justifique.

c. ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la *UID* de *root*?
d. Agregue un nuevo usuario llamado *isocso* a su instalación de GNU/Linux, especifique que su home sea creada en /home/*isocso*, y hágalo miembro del grupo *informatica* (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.
e. Investigue la funcionalidad y parámetros de los siguientes comandos:

i. useradd y adduser
ii. usermod
iii. userdel
iv. su
v. groupadd
vi. who
vii. groupdel
viii. passwd

16. FileSystem y permisos:
a. ¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?
b. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:
    i. chmod
    ii. chown
    iii. chgrp
c. Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?
d. ¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Indiquelo y realice las pruebas correspondientes.
e. Explique los conceptos de “full path name” (path absoluto) y “relative path name” (path relativo). De ejemplos claros de cada uno de ellos.
f. ¿Con qué comando puede determinar en qué directorio se encuentra actualmente? ¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo.
g. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

i. umount
ii. du
iii. df
iv. mount
v. mkfs
vi. fdisk (con cuidado)
vii. write
viii. losetup
ix. stat

17. Procesos:
    a. ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?
    b. ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?
    c. Pipe (|). ¿Cuál es su finalidad? Cite ejemplos de su utilización.
    d. Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.

18. Otros comandos de Linux (Indique funcionalidad y parámetros):
    a. ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?
    b. Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué característica nota?
    c. ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados.
    d. ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?
    e. Investigue la funcionalidad de los siguientes comandos:

        i. tar
        ii. grep
        iii. gzip
        iv. zgrep
        v. wc

19. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando, indique la razón:

```
l s –l > prueba
ps > PRUEBA
chmod 710 prueba
chown root:root PRUEBA
chmod 777 PRUEBA
chmod 700 /etc/passwd
passwd root
rm PRUEBA
man /etc/shadow
find / –name * .conf
usermod root –d /home/ newroot –L
cd / root
rm *
cd / etc
cp * /home –R
shutdown
```

20. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:

a. Terminar el proceso con *PID* 23.
b. Terminar el proceso llamado *init* o *systemd*. ¿Qué resultados obtuvo?
c. Buscar todos los archivos de usuarios en los que su nombre contiene la cadena ".conf"
d. Guardar una lista de procesos en ejecución el archivo **/home/<su nombre de usuario>/procesos**
e. Cambiar los permisos del archivo **/home/<su nombre de usuario>/xxxx** a:
    i. Usuario: Lectura, escritura, ejecución
    ii. Grupo: Lectura, ejecución
    iii. Otros: ejecución
f. Cambiar los permisos del archivo **/home/<su nombre de usuario>/yyyy** a:
    i. Usuario: Lectura, escritura.
    ii. Grupo: Lectura, ejecución
    iii. Otros: Ninguno
g. Borrar todos los archivos del directorio **/tmp**
h. Cambiar el propietario del archivo **/opt/isodata** al usuario **isocso**
i. Guardar en el archivo **/home/<su nombre de usuario>/donde** el directorio donde me encuentro en este momento, en caso de que el archivo exista no se debe eliminar su contenido anterior.

21. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:
    a. Ingrese al sistema como usuario "root"
    b. Cree un usuario. Elija como nombre, por convención, la primera letra de su nombre seguida de su apellido. Asígnele una contraseña de acceso.
    c. ¿Qué archivos fueron modificados luego de crear el usuario y qué directorios se crearon?
    d. Crear un directorio en /tmp llamado *miCursada*
    e. Copiar todos los archivos de /var/log al directorio antes creado.
    f. Para el directorio antes creado (y los archivos y subdirectorios contenidos en él) cambiar el propietario y grupo al usuario creado y grupo users.
    g. Agregue permiso total al dueño, de escritura al grupo y escritura y ejecución a todos los demás usuarios para todos los archivos dentro de un directorio en forma recursiva.
    h. Acceda a otra terminal para loguearse con el usuario antes creado.
    i. Una vez logueado con el usuario antes creado, averigüe cuál es el nombre de su terminal.
    j. Verifique la cantidad de procesos activos que hay en el sistema.
    k. Verifiqué la cantidad de usuarios conectados al sistema.
    l. Vuelva a la terminal del usuario root y envíele un mensaje al usuario anteriormente creado enviándole que el sistema va a ser apagado.
    m. Apague el sistema.

22. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes

acciones:

n. Cree un directorio cuyo nombre sea su número de legajo e ingrese a él.
o. Cree un archivo utilizando el editor de textos *vi*, e introduzca su información personal: Nombre, Apellido, Número de alumno y dirección de correo electrónico. El archivo debe llamarse "LEAME".
p. Cambie los permisos del archivo LEAME, de manera que se puedan ver reflejados los siguientes permisos:
   - ➢ Dueño: ningún permiso
   - ➢ Grupo: permiso de ejecución
   - ➢ Otros: todos los permisos
q. Vaya al directorio /etc y verifique su contenido. Cree un archivo dentro de su directorio personal cuyo nombre sea **leame** donde el contenido del mismo sea el listado de todos los archivos y directorios contenidos en /etc. ¿Cuál es la razón por la cuál puede crear este archivo si ya existe un archivo llamado "LEAME" en este directorio?
r. ¿Qué comando utilizaría y de qué manera si tuviera que localizar un archivo dentro del filesystem? ¿Y si tuviera que localizar varios archivos con características similares? Explique el concepto teórico y ejemplifique.
s. Utilizando los conceptos aprendidos en el punto anterior, busque todos los archivos cuya extensión sea .so y almacene el resultado de esta búsqueda en un archivo dentro del directorio creado en el primer inciso. El archivo deberá llamarse *ejercicioF*.

23. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando indique la razón:

```
01.  mkdir iso
02.  cd . / iso; ps > f0
03.  ls > f1
04.  cd /
05.  echo $HOME
06.  ls -l $> $HOME/ iso/ls
07.  cd $HOME; mkdir f2
08.  ls -ld f2
09.  chmod 341 f2
10.  touch dir
11.  cd f2
12.  cd ~/iso
13.  pwd > f3
14.  ps | grep 'ps' | wc -l >> ../f2/f3
15.  chmod 700 ../f2 ; cd ..
16.  find . -name etc/passwd
17.  find / -name etc/passwd
18.  mkdir ejercicio5
19.  . . . . . .
20.  . . . . . .
```

a. Inicie 2 sesiones utilizando su nombre de usuario y contraseña. En una sesión vaya

siguiendo paso a paso las órdenes que se encuentran escritas en el cuadro superior. En la otra sesión, cree utilizando algún editor de textos un archivo que se llame “explicacion_de_ejercicio" dentro del directorio creado en el ejercicio 22 y, para cada una de los comandos que ejecute en la otra sesión, realice una breve explicación de los resultados obtenidos.

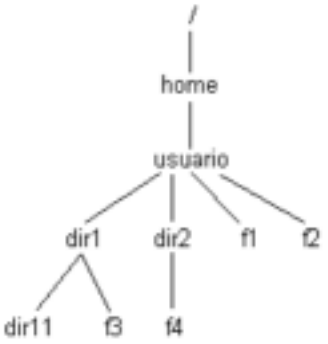
b. Complete los comandos 19 y 20, de manera tal que realicen la siguiente acción:

   19: Copiar el directorio iso y todo su contenido al directorio creado en 24.a

   20: Copiar el resto de los archivos y directorios que se crearon en este ejercicio al directorio creado en el ejercicio 24.a

c. Ejecute las órdenes 19 y 20 y coméntelas en el archivo creado en el inciso a).

24. Cree una estructura desde el directorio /home que incluya varios directorios, subdirectorios y archivos, según el esquema siguiente.



Asuma que “usuario” indica cuál es su nombre de usuario. Además deberá tener en cuenta que dirX hace referencia a directorios y fX hace referencia a archivos. Utilizando la estructura de directorios anteriormente creada, indique qué comandos son necesarios para realizar las siguientes acciones:

a. Mueva el archivo "f3” al directorio de trabajo /home/usuario.
b. Copie el archivo "f4” en el directorio "dir11".
c. Haga los mismo que en el inciso anterior pero el archivo de destino, se debe llamar "f7".
d. Cree el directorio copia dentro del directorio usuario y copie en él, el contenido de "dir1".
e. Renombre el archivo "f1" por el nombre *archivo* y vea los permisos del mismo.
f. Cambie los permisos del archivo llamado *archivo* de manera de reflejar lo siguiente:
   - ➢ Usuario: Permisos de lectura y escritura
   - ➢ Grupo: Permisos de ejecución
   - ➢ Otros: Todos los permisos
g. Renombre los archivos "f3” y “f4" de manera que se llamen "f3.exe” y “f4.exe” respectivamente.
h. Utilizando un único comando cambie los permisos de los dos archivos renombrados en el inciso anterior, de manera de reflejar lo siguiente:
   - ➢ Usuario: Ningún permiso

- ➢ Grupo: Permisos de escritura
- ➢ Otros: Permisos de escritura y ejecución

25. Indique qué comando/s es necesario para realizar cada una de las acciones de la siguiente secuencia de pasos (considerando su orden de aparición):
    a. Cree un directorio llamado *logs* en el directorio /tmp.
    b. Copie todo el contenido del directorio /var/log en el directorio creado en el punto anterior.
    c. Empaquete el directorio creado en a), el archivo resultante se debe llamar "misLogs.tar".
    d. Empaquete y comprima el directorio creado en a), el archivo resultante se debe llamar "misLogs.tar.gz".
    e. Copie los archivos creados en c) y d) al directorio de trabajo de su usuario.
    f. Elimine el directorio creado en a), *logs*.
    g. Desempaquete los archivos creados en c y d en 2 directorios diferentes.