



Final

Apellido y Nombres	Nro de Legajo	Profesor

Normas Generales

Lea detenidamente cada pregunta y consulte las dudas de interpretación que pudiesen surgir.

Sección Teórica (tiempo estimado 30 minutos):

Punto 1) Si la máquina se llama **INFO-UX**, la fecha del equipo es **15-MAY-2021**, el usuario conectado en el equipo es **Luis Aceto** (ID de usuario **LACETO**) y el directorio en el que se encuentra en ese momento es **INFO_1/Clase2** dentro del directorio personal, ¿cuál es el prompt que mostrará en la consola? (Se considera el prompt que configura el Sistema Operativo por defecto). Seleccione una opción:

- A. INFO-UX:/home/LACETO/INFO_1/Clase2\$
- B. Luis.Aceto@INFO-UX:/home/Luis.Aceto/INFO_1/Clase2\$
- C. Luis.Aceto@INFO-UX:~/INFO_1/Clase2\$
- D. LACETO@INFO-UX:~/INFO_1/Clase2\$
- E. 2021-05-15@INFO-UX:~/INFO_1/Clase2\$
- F. INFO-UX@LACETO:~/INFO_1/Clase2\$
- G. Ninguna es correcta.

Punto 2) Indique cuáles de las siguientes afirmaciones son verdaderas:

- A. Linux maneja los dispositivos como si fueran un archivo y el encargado de hacerlo para cada dispositivo es el **device driver o controlador del dispositivo**.
- B. Una desventaja del método de complemento a 1 para la representación de números signados es que tengo un doble cero.
- C. Todos los procesos tienen un solo proceso padre y pueden ser padres de un solo proceso hijos a la vez. Si quiero tener un segundo proceso hijo, debo esperar a que finalice el primero con la función **wait**.
- D. **Para generar un ejecutable** a partir de un archivo fuente en C **debo realizar dos pasos**. El primero es la **compilación** y el segundo es el **linkeo o enlace**.
- E. Para generar un programa en C puedo usar un procesador de texto como el word si me aseguro de no ponerle ningún formato como cambio de tamaño de letra o negrita.
- F. Los archivos en Linux manejan tres grupos de permisos, los correspondientes al dueño, a un grupo y al resto de los usuarios.
- G. No se puede establecer una conexión a través de un socket si ambos procesos se ejecutan en la misma máquina.
- H. El sistema de numeración está definido por la cantidad de símbolos que utiliza. Esta cantidad se llama **BASE**.
- I. El **named pipe o FIFO** es un método de IPC que permite vincular dos procesos que se ejecuten en la misma máquina aunque no estén relacionados como padre-hijo.
- J. La ventaja de trabajar con el sistema **HEXADECIMAL** es que la conversión desde binario es directa a partir de la agrupación de **tres dígitos binarios**.
- K. En el caso de Linux, la interfaz gráfica la puedo cambiar e incluso eliminarla.
- L. Los **procesos se identifican con un número**, y una vez que se asignó el número mayor, hay que reiniciar el equipo para volver el contador a 0.
- M. Si la máquina tiene la suficiente capacidad, puedo tener dos Sistemas Operativos controlando el hardware simultáneamente.

Apellido y Nombres	Nro de Legajo

- N. El **Sistema Operativo** se conecta con el hardware a través de las **system calls**.
 O. En una conexión vía **Socket**, cuando un proceso **servidor quiere aceptar la comunicación** con un cliente, debe usar la función **connect**.

Punto 3) ¿A que se denomina Kernel de un sistema operativo Linux? Seleccione una opción:

- A. A la interfaz que permite al usuario ingresar comandos.
 B. A la lista de comandos externos que pueden ser ejecutados por el sistema operativo.
 C. Al conjunto de programas que constituyen el Sistema Operativo propiamente dicho y a su correspondiente implementación de las System Calls.
 D. Al conjunto de políticas que definen la manera en que se almacenan los archivos en el disco rígido.
 E. Ninguna es correcta.

Nota para la parte teórica:

Se deberá entregar un archivo de texto donde indique el punto y la o las letras de las respuestas que considere correctas.

Para aprobar la parte teórica deberá tener mínimo 9 puntos.

Para los puntos 1 y 3, respuesta correcta suma uno, respuesta incorrecta resta 1. No respondida, no suma ni resta.

Para el punto 2 cada respuesta indicada se toma como verdadera, la no marcada se toma como falsa. Cada respuesta correcta suma uno, cada respuesta incorrecta resta 1.

Sección Práctica (tiempo estimado 2 horas):

Punto 1) Realizar una función que reciba un buffer de enteros y los convierta a un string.

Considerar que los enteros son de 4 bytes y los valores correspondientes al **código ASCII son los 8 bits del segundo byte menos significativo**, es decir los 16 bits de mayor peso no importan, los 8 bits que siguen son el código ASCII y los 8 bits de menor peso no importan.

Definir el prototipo que considere más conveniente sabiendo que **debe recibir un vector de enteros, su longitud y devolver un string**. También deberá tener la opción de **retornar un código de error en caso de no poder generar la frase**.

Para probar la función puede definir el vector:

`int v[4]={147456,227328,221184,198656};`

que dará como resultado la frase **"Hola"**.

Punto 2) Realizar una función que reciba un string correspondiente a un nombre de archivo, busque el último punto a la derecha y devuelva un string que corresponda al nombre original, agregada la frase **"_decrypt"** y la extensión.

Ejemplo: Recibe el nombre **"texto.txt"** y debe retornar **"texto_decrypt.txt"**

Si el nombre no contiene punto, la frase se agregará al final y quedará sin punto.

El prototipo de la función deberá ser:

`int nvo_nombre (char *, char *);`

Donde el **primer parámetro es el nombre del archivo original, el segundo es el nombre del archivo nuevo y retorna un OK o ERROR** (dos constantes simbólicas que deberá definir como 0 y 1) si lo pudo procesar o no.

Punto 3) Realizar una función que reciba el nombre de un archivo. El archivo es del tipo binario y contiene una sucesión de números enteros de 4 bytes. No se conoce previamente su longitud.

La función deberá leer el archivo número por número e ir guardando los valores en un buffer de enteros que será devuelto como resultado de la función a través de una argumento pasado por referencia.

El prototipo de la función deberá ser:

Apellido y Nombres	Nro de Legajo

*int lectura (char *, int **);*

Donde el **primer parámetro es el nombre del archivo, y el segundo es el lugar para guardar la dirección de memoria donde comienza el buffer. Retornará un número mayor o igual a cero con la cantidad de elemento del vector o ERR_LEC** (constante simbólica que deberá definir como -1) si no lo pudo generar.

Punto 4) Realizar un programa que reciba por argumento de main uno o varios nombres de archivos. Por cada nombre deberá generar un proceso hijo que se encargará de abrirlo y cargarlo en un buffer utilizando la función del punto 3.

Cada proceso deberá convertir el contenido del buffer a texto utilizando la función del punto 1 y generar un nuevo archivo (de texto) con la frase convertida, cuyo nombre sea el generado en la función 2.

Para probar la función main, si no tiene desarrollada la función del punto 1, puede utilizar un string cualquiera en reemplazo de la función.

Para probar la función main, si no tiene desarrollada la función del punto 2, puede utilizar un nombre de archivo cualquiera en reemplazo de la función.

Deberá validar todos los errores posibles presentando por pantalla el mensaje informativo correspondiente en caso de producirse alguno.

El proceso padre, a medida que vaya disparando los hijos, deberá ir completando un vector de estructuras con la información de cada proceso. El tamaño máximo de este vector será de 10 elementos.

Una vez que disparó todos los hijos y completó el vector, quedará en espera.

Finalizará cuando hayan terminado todos los hijos asegurándose que hayan concluido correctamente.

La estructura que utilizará para almacenar la información, será:

```
struct hijos
{
    int pid;           //Número del proceso hijo generado.
    char * archivo;    //Nombre del archivo que utiliza dicho proceso.
    int nro_arg;       //Número de orden en el que fue cargado el nombre en la línea de comando.
};
```

Nota para la parte práctica:

Deberá entregar tres archivos fuentes. El main.c donde sólo conste la función main. El funciones.c donde se encuentren el resto de las funciones pedidas y el funciones.h donde están los prototipos y toda aquella información adicional que considere oportuna.

Recuerde aplicar las buenas prácticas de programación para el uso de archivos y memoria dinámica.

La valoración de los puntos 1, 3 y 4 es de 3 puntos cada uno y 1 punto para el ítem 2.

Además de alcanzar el umbral de 6 puntos deberá tener 2 funciones bien resueltas y 1 bien planteada o 1 bien resueltas y 3 bien planteadas.