Indique y explique 3 criterios a tener en cuenta para optar por una implementación multi-proceso o por una multi-hilo.

Primer criterio:
Estimar la necesidad y problemas de contar con procesos ya sea en paralelo o concurrente según la funcionalidad del trafico
Segundo criterio :
Obligación de compartir los mismos recursos/datos si es necesario de un proceso o evaluar si no deberían de compartirlos.
Tercer criterio:
Uniformidad de las maniobras solicitadas/demandadas, para tener varios procesos de distintos programas osino varios hilos en un programa.

Explique al menos un uso programático para el valor de "Process Id" de un proceso. Si lo desea, puede realizar una explicación proporcionando un ejemplo.

Un uso programático seria el de pedir el valor del Process Id para saber si se trata de un proceso hijo o del proceso padre sabiendo que cada uno arroja un valor predeterminado, el hijo arroja 0 y el padre un valor aleatorio que puede ser hasta mas grande de 650000

Explique brevemente en sus términos el uso apropiado de las funciones fork y exec en el ámbito de la programación de aplicaciones multi-proceso.

Los términos fork y exec se relacionan normalmente en conjunto porque en fork creamos un proceso hijo, en donde se realiza una copia en memoria de trabajo del proceso original al que lo denominaremos padre y exec para crear un nuevo programa en ese espacio creado. A través de esto se logra que un mismo programa pueda correr subprogramas como procesos únicos e independientes pero con la misma funcionalidad de negocio para sacar beneficio de la capacidad de multiproceso del hardware de nuestro computador

¿Cuál es la diferencia entre una compilación con enlace dinámico y otra con enlace estático?. Según su criterio, indique al menos una ventaja que se deriva en cada caso.

La vinculación estática se realiza en tiempo de compilación mediante una herramienta llamada enlazador mientras que la vinculación dinámica se realiza en tiempo de ejecución, por el sistema operativo.
Una ventaja Con el enlace estático seria que el compilador puede optimizar el código de la biblioteca, pero solo si también lo compila Si solo se vincula a archivos de objetos precompilados, su compilador no tiene la oportunidad de optimizarlos.La vinculación estática también puede hacer que los archivos binarios sean más fáciles de distribuir a diversos entornos de usuario (a costa de enviar un programa más grande y con más recursos).

Ventaja del enlace dinámico puede reducir el consumo total de recursos (si más de un proceso comparte la misma biblioteca (incluida la versión en "la misma", por supuesto)).
el enlace dinámico tiene la ventaja que las correcciones de errores y las actualizaciones a las bibliotecas se propagan para mejorar su producto sin requerir que envíe nada.

Indique brevemente a que ser refiere el valor de niceness de un proceso. ¿En que caso o escenario usted considera que este valor debería ser modificado ?

el valor de niceness se refiere al valor de precedencia o preferencia que se le da a un proceso con respecto a los demás. Se debería cambiar cuando sea necesario cierto proceso decisivo, otorgándole un valor de niceness bajo, o otorgándole un valor de niceness alto a un proceso no tan imperioso en nuestro código.

✗ Explique en sus propios términos a que se refiere el fenómeno de "condición de carrera" en la programación multihilo y por que debe de ser evitada. 1/2

La condición de carrera ocurre cuando varios subprocesos leen y escriben la misma variable, es decir, tienen acceso a algunos datos compartidos e intentan cambiarlos al mismo tiempo. En este momento, los hilos están luchando entre ellos para acceder/ modificar los datos. Debería ser evitada porque genera problemas de vulnerabilidad de seguridad importante.

Explique a que se refieren las "Señales" en el contexto de Procesos Pesados. Si lo desea, puede realizar una explicación proporcionando un ejemplo.

Las señales en contexto de procesos pesados serian mensajes que envía el sistema operativo a los procesos y que pueden lograr modificar su ejecución natural, hasta podría modificar también la situación de los recursos utilizados por el programa.

Indique en sus términos la importancia de aplicar un enfoque de Programación Defensiva desde el punto de vista de las fallas de Sistemas.

Las fallas normalmente se producen en los momentos de ejecución, que puede ser por manejo incorrecto o por problemas imprevistos. Es por eso que la programación defensiva es importante ya que nos ayuda gratamente al dejarnos que nuestro programa se ejecute realizando su funcionamiento mediante el llamado al sistema necesario, ademas de la generación de la documentación y aclaraciones que permitirá la corrección de los errores y fallas con anticipación y a futuro.

Indique la diferencia entre Mutex y Semáforos en un contexto de programación multi-hilo. Explique al menos un caso de aplicación de cada uno.

MutEx se usa para limitar o cancelar el acceso a una sección decisiva y fijandolo de un hilo a la vez(que seria la condición de carrera), mientras que los semáforos son contadores que se usan para sintonizar varios hilos mediante otro hilo. Ejemplos:
En un local de tienda de ropas pueden entrar hasta 10 personas, en este caso usaríamos el semáforo porque el semáforo permite que varios threads accedan a un mismo recurso hasta un numero máximo y el mutex podría ser el caso de un vestidor dentro del local en el cual solo puede ingresar una sola persona al mismo tiempo.

✓ Explique a que se refiere una "Sección Crítica no Cancelable" en un contexto de programación multi-hilo. 2/2

La Sección critica no cancelable hace referencia a un pedazo del programa que debe ser conformado por un solo hilo por vez mediante una restricción que no pueda ser anulado obligadamente por el llamado del sistema.

(a) Indique la diferencia entre variables locales y variables globales. (b) ¿Cómo pueden identificarse/diferenciarse las mismas?. (c) Indique como ejemplo un comando que consulta el valor de la variable HOSTNAME.

a)Las variables locales solo pueden verse en su propio contexto mientras que las variables globales son las que se caracterizan por los diferentes programas y subprogramas de estos.
b)Las locales pueden ser observadas con el comando env y printenv.
Las variables globales pueden ser observadas con el comando set
c-) echo "El valor de Hostname es:$HOSTNAME"

✕ Explique por qué una sesión Bash no termina o finaliza cuando se aplica la combinación Ctrl + C 0/1

Porque el bash no se encuentra ejecutando ningún programa.

Escriba un script en el que se asigna programáticamente valores a dos variables para luego, calcular el perímetro y el área de un rectángulo con esas proporciones. Su script debe estar comentado y debe producir una salida "elegante"

```
#!/bin/bash

#Limpieza de la consola
clear

echo*
#Inicializamos las variables y asignamos valores int,
echo "Inicializando 2 variables"
VARIABLE1="5"
VARIABLE2="6"
echo
echo "Variables creadas y valores de las dimensiones asignadas"


#Mostramos el contenido de las variables
echo "Mostramos el valor de las variables"
echo
echo "VARIABLE1 es igual a $VARIABLE1 "
echo "VARIABLE2 es igual a $VARIABLE2"
echo
echo "LA VARIABLE1 ES EL LARGO Y VARIABLE2 ES EL ANCHO"


#Calculamos y mostramos el perimetro y area del rectangulo
echo "Calculando el perimetro del rectangulo"
PERIMETRO=$(($VARIABLE1+$VARIABLE2))
echo
echo "El Perimetro del rectangulo es:$(($PERIMETRO*2))"
echo
echo "Calculando el Area del rectangulo"
echo
echo "El Area del rectangulo es: $(($VARIABLE1*$VARIABLE2))"
```

Que entiende por la primera línea de un archivo shell script que normalmente es de la forma: #!/bin/bash

De la primera linea de comando #!/bin/bash entiendo que es un comando el cual se utiliza para indicar el tipo de shell en este caso el bash, que sera utilizado para ejecutar el script, el bash es un interprete/traductor de comandos que esta inspirado en UNIX

A su criterio, ¿existe alguna diferencia programática o de efecto al correr un script en modo normal o en modo debug?. Indique que es posible al correr un script en modo debug.

Si existe, para mi las diferencias están que al ejecutar un script en modo normal el programa corre y se ve en consola solamente aquellos comandos que manden una salida en consola como por ejemplo el echo mientras que en el modo debug podemos ejecutar el programa línea a línea, para comprobar su correcto funcionamiento.Lo cual es super eficiente para encontrar los errores en tu código.

Explique los permisos que deben otorgarse a un archivo script para que el mismo pueda ejecutarse.

Los permisos deben ser otorgados para que el script pueda ejecutarse se realiza por medio del comando: chmod u+x

Normalmente, cuando el shell inicia, el mismo lee sus archivos de configuración. Indique las opciones que representan algunos de estos archivos.

- [ ] /etc/sh/
- [x] ~/.bash_profile
- [x] ~/.bashrc
- [ ] /etc/rc.local

No hay respuestas correctas

Explique a su mejor entender la importancia de contar con un Shell en un sistema operativo.

A mi entender la caracteristica mas importante por la cual hay que contar con un Shell en un sistema operativo es porque un Shell es necesario para invocar o ejecutar los distintos programas disponibles en la computadora, es decir sin ellas no podremos ejecutar ningun solo programa. Tambien se encarga de gestionar la relacion entre usuario/sistema operativo pidiendo la entrada, luego traduce/interpreta la entrada para el sistema operativo y la salida

Indique la ubicación en el sistema de archivos donde (normalmente) se ubica el programa interpreta Bash del sistema.

/etc/passwd

Que se entiende por las siglas POSIX. Indique un ejemplo de ello en programación Bash.

Se entiende por las siglas POSIX del ingles,Portable Operating System Interface for uni-X, que seria la interfaz de sistema operativo portable. El bash traduce los scripts a traves del shell POSIX.

```
#!/bin/bash
testfuncion
testfuncion(){
  echo "Mi primera funcion"
}
```

✓ Describa brevemente a su mejor entender la importancia de las expresiones regulares para el procesamiento de cadenas. 1/1

Es importante ya que son de gran utilidad para filtrar cadenas de texto en donde se buscan ciertos patrones de combinación de caracteres

✓ Escriba un comando que liste todos los archivos del directorio /usr/bin que tengan una letra “a” como segundo carácter y que guarde el resultado en un archivo .txt ubicado en el directorio /tmp 1/1

ls /usr/bin | sed -n '/^.a/p' > /tmp/archivo.txt

✓ Escriba un script bash que indique si un nombre de usuario recibido como parámetro se encuentra o no listado en el archivo /etc/passwd. 1/1

```
#Cargamos en var la variable del usuario, en este caso Juan
VAR=Juan
#Corroborar si un usuario se encuentra en el listado
echo "Corroborando si el usuario $VAR se encuentra en el listado.."
echo " Para corroborar usaremos la impresion del 1 si se encuentra en el listado y sino  0 sino se encuentra.:"
cat /etc/passwd | grep ^"$VAR:" -c
```

✓ Resuelva el siguiente planteamiento: 1/1

Escriba un script con awk que convierta una línea en este formato:

```
Username:Firstname:Lastname:Telephone number
```

en un registro con este formato:

```
dn: uid=Username, dc=example, dc=com
cn: Firstname Lastname
sn: Lastname
telephoneNumber: Telephone number
```

```
awk -f: '{print "dn:uid = "$1",dc =example,dc = example, dc= com ";print "cn: "$2 $3; print "sn: " $3; print "telephoneNumber: ""$4;print" }' file
```

✓ Escriba un comando que liste por pantalla todos los usuarios del sistema que inician sesión con Shell Bash por defecto. 1/1

grep bash /etc/passwd | cut -d: -f1

✓ Describa brevemente la utilidad de los comandos: grep, sed y awk. 1/1

grep: es una herramienta de linea de comando que sirve para la búsqueda de patrones en cadenas
sed: es un editor de flujo que se usa normalmente para buscar y sustituir cadenas en un texto
awk: es un lenguaje de programación fácil de aprender para procesar datos basados en texto

Test 3 - Programación Shell

Puntos totales 4/4

Test 3 correspondiente al tópico de Programación Shell
==========
1. Realice el cuestionario en forma individual.
2. Dispone de 1 hora de tiempo, el cuestionario cierra a las 21.00.
3. Evite enviar sus respuestas con retraso, el formulario puede cerrarse evitando que pueda responder.
4. Puede utilizar todos los materiales a su disposición.

Se ha registrado el correo del encuestado (**martin.aponte8@fpuna.edu.py**) al enviar este formulario.

✓ Escriba un script denominado testdaemon.sh que compruebe si un servicio (demonio) cuyo nombre es recibido como parámetro se encuentra corriendo o no. 1 / 1

```
#! /bin/bash
#Buscamos el daemon en el servidor, por ejemplo en httpd
SERVICE = httpd
#hacemos un if buscando si el servicio esta corriendo, si esta imprimimos que lo esta, sino
#imprimimos que no esta corriendo
if ps ax | grep -c grep | grep $SERVICE >/dev/null
then
  echo "El daemon $SERVICE se encuentra corriendo de manera correcta"
else
  echo "El daemon $SERVICE no se encuentra corriendo..."
fi
```

✓ Escriba un script que tome exactamente un argumento, un nombre de directorio, tal que: (a) Si el número de argumentos es diferente de uno, imprima un mensaje de uso. (b) Si el argumento no es un directorio, que imprima un mensaje de error informativo. (c) Para el directorio dado, liste los cinco archivos más grandes primeramente y luego, los cinco archivos que fueron modificados más recientemente. 1 / 1

```
#!/bin/bash

# $1indica el 1er argumento
argumento=$1

# comprobamos en la lista se da un argumento
if [ ! $# == 1 ];
then
echo "Obs: Utd puede enviar sólo un directorio como argumento."
 exit
fi

# comprobamos que el argumento es un directorio
if [ -d "${argumento}" ];
# es un directorio
then
 echo "A continuación se enumeran los cinco archivos más grandes"
 du -ah "${argumento}" | sort -rh | head -n 5

 echo "A continuación se enumeran los cinco archivos que se modificaron más recientemente"
 ls -ltr | tail -n 5

else
# no es un directorio.
echo "Obs: tu argumento no es un directorio"
fi
```

✓ Escriba un script que solicite el ingreso por teclado de la edad del usuario. Si el valor es igual o superior a 20, debe imprimir un mensaje diciendo que el usuario puede beber alcohol. Si la edad del usuario es inferior a 20 años, imprima un mensaje indicándole cuántos años debe esperar como máximo para beber legalmente. 1 / 1

```
#! bin/bash
#consultamos la edad de la persona
echo "Ingrese su edad por favor:"
#leemos la edad
read EDAD
#si la edad es igual o superior a 20 imprimimos que puede beber
if[$EDAD -ge 20];
then
  echo "Puede beber alcohol"
else
  ANO = $(echo "scale=2; $(($EDAD*365))"|bc -l)
  DIAS = $(echo "scale=2; $((20*365 -$ANO))"|bc -l)
fi
```

✓ Utilice una construcción if / then / elif / else que imprima información sobre el mes actual. El script debe indicar el número de días del mes actual e indicar si el año es bisiesto. 1 / 1

```
#declaramos las variables de año mes y var
año=$(date +%Y)
mes=$(date +%m)
var=$(date +%B)

# If para ver los meses que poseen 31 días, como octubre que es el mes actual
if (( $mes == 12 )) || (( $mes == 10 )) || (( $mes == 8 )) || (( $mes ==7 )) || (( $mes == 5 )) || ((
$mes == 3 )) || (( $mes == 1 ));
then
echo "$var tiene 31 días"
echo "El año en este caso no es bisiesto "

#en caso de no ser los meses del if anterior y del caso especial de febrero seran 30 dias
elif (($mes != 2 ));
then
echo "$var tiene 30 dias,  el año en este caso bisiesto "
echo "El año en este caso no es bisiesto "

#en caso de ser febrero verificamos si es bisiesto o no, si es seran 29 dias sino 28
elif (( $año % 400 == 0)) || ( (($año % 4 == 0)) && (( $año % 100 !=0)) );
then
echo "$var tiene 29 días ya que es un año bisiesto"

#en caso de no ser bisiesto tendrá 28 días
else
echo "$var tiene 28 dias, el año en este caso no es bisiesto"
fi
```

Test 4 - Programación Shell

Puntos totales 4/4

Test 4 correspondiente al tópico de Programación Shell
==========
1. Realice el cuestionario en forma individual.
2. Dispone de 1 hora de tiempo, el cuestionario cierra a las 21.00.
3. Evite enviar sus respuestas con retraso, el formulario puede cerrarse evitando que pueda responder.
4. Puede utilizar todos los materiales a su disposición.

Se ha registrado el correo del encuestado (**martin.aponte8@fpuna.edu.py**) al enviar este formulario.

✓ Escriba un script bash que: (a) Indique el nombre del script, (b) Indique cuantos argumentos ha recibido y (c) Imprima los argumentos recibidos en orden inverso. 1 / 1

```
#!/bin/bash

# a) Indicamos el nombre del script con basename
echo "`basename "$0"`"

#b) Indicamos cuantos argumentos se han recibido con $#
echo "$#"

# c) Imprimimos los argumentos recibidos en orden inverso
echo $@ | tr ' ' '\n' | tac | tr '\n' ' '
```

✓ Cree un script que invoque al comando "dd" para crear una imagen de arranque. Para el mismo, si el usuario intenta interrumpir la secuencia de comandos mediante la combinación "Ctrl + C", se muestre un mensaje indicando que esta acción hará que el proceso terminará y que el disquete quede no utilizable. 1 / 1

```
#!/bin/bash

   #Vamos a crear un script donde invocamos a dd para crear una imagen y lo guardamos

    trap "echo Si quieres cortar el proceso ahora, lo terminaras y el disquete quedara inutilizado" SIGNINT
#utilizamos el comando dd para crear y guardarlo
dd if=/dev/sda6 conv=sync,noerror bs=128K | gzip -c > imagendearranque.gz
```

✓ Explique la finalidad del mecanismo de Signals y Traps en Programación Bash. Indique un ejemplo que ilustre el caso. 1 / 1

La finalidad del mecanismo Signals es la de enviar mensajes bloqueantes que se envían a un programa para advertir de un evento importante, mientras que la finalidad el mecanismo de Traps es aquella que a traves de funciones captura un signal y evita que este se ejecute.
#Ejemplo que ilustre el caso. Seria el que usamos en la imagen, para el mensaje de #advertencia

```
#!/bin/bash
#
 trap "echo Si quieres cortar el proceso ahora, lo terminaras y el disquete quedara inutilizado" SIGNINT
```

✓ Resuelva el siguiente enunciado: 1/1

Escriba un script bash denominado **service** que soporte la recepción de parámetros de la siguiente forma **service <nombre_demonio> <accion>** y cuya implementación esté basada en funciones que implementen los comandos de acción sobre un demonio de: inicio (start), detenimiento (stop) o reinicio (restart).

Considere que un demonio puede ser iniciado con el siguiente comando:

su -c "nohup demonio >> /var/log/demonio.log &"

que retorna el PID de proceso que luego debe ser guardado en un archivo de la forma **/proc/run/demonio.pid** para su uso futuro (por ejemplo para invocar al comando **kill**).

```
#!/bin/bash
#Usaremos como Parametro posicional DEMONIO= $1
ACC=$2
ARCGHIVOPID="/proc/run/demonio.pid"
start() {
        DEMONIO=$1
        #en $2 ponemos el archivopid
        ARCHIVOPID = $2
        PID = su -c "nohup $DEMONIO >> /var/log/demonio.log&"
        $PID > $ARCHIVOPID
        #comenzamos el demonio
        echo " Ha iniciado el Demonio $DEMONIO sin inconvenientes"
}
restart() {
        #reiniciamos el demonio usando primero stop y luego start
        stop $1 $2
        start $1 $2
        # imprimimos que se ha reiniciado el demonio
        echo " Ha reiniciado el Demonio $DEMONIO de manera correcta"
}
stop() {
        PID = "cat $2"
        kill $PID
        >$2
       #detenemos el demonio con el comando kill
        echo " Ha detenido el Demonio $DEMONIO de manera exitosa"
}

#los diferentes casos según la acción
case $ACC in
#caso 1 de comenzar
COMENZAR)
    start $DEMONIO $ARCHIVOPID
    ;;
#caso 2 de reiniciar
REINICIAR)
    restart $DEMONIO $ARCHIVOPID
    ;;
#caso 3 de detener
DETENER)
   stop $DEMONIO $ARCHIVOPID
    ;;
*)
#lanzamos tambien un mensaje de error en caso de escribir una opcion erronea
  echo "Error. Pruebe COMENZAR, DETENER o REINICIAR por favor"
   ;;
esac
```