Trabajo de Diseño y Administración de Sistemas Operativos

Alumno: Noelia Osta Supervia
DNI: 17768378-G
Centro Asociado: Calatayud
Teléfono de contacto: 644417668
Email: nosta1@alumno.uned.es

Primera PED

o Introducción

En este apartado se introduce muy brevemente lo que se ha hecho en el trabajo.

Tal y como requería la PED, se ha codificado en bash el script Demonio y el script Fausto, para lograr la simulación de los procesos demonio.

Fausto. Shell script que recibe órdenes por línea de comandos e invoca al demonio cuando no hay un demonio en ejecución. Según el comando, ejecutara unas acciones o otras.

Demonio. Se encuentra en un bucle infinito leyendo listas y realizando las tareas pertinentes hasta que Fausto recibe la orden de apocalipsis y por tanto, crea ese archivo. El demonio entonces, comenzará el apocalipsis hasta detener por completo su ejecución.

Ambos scripts se empezaron a implementar en la máquina virtual, pero, por diversos problemas (no poder compartir archivos, una máquina virtual no me volvió a arrancar, perdiendo todo lo realizado en la práctica, etc) decidí instalar una partición de kubuntu en mi ordenador e instalar en ella la maquina virtual (la cual extrañamente me funciona mucho mejor que en Windows, pudiendo compartir archivos sin problema y sin tener problemas). Por tanto, para la realización de esta práctica se ha usado la MV, un SO kubuntu, como editor de archivos kate, y onlyoffice para la redacción de este informe.

o Implementación

Aquí se describirá el programa implementado y los resultados obtenidos.

Fausto.sh

Descripción.

Shell script que recibe órdenes por la linea de comandos, los guarda como argumentos $n (siendo n un numero comprendido entre 0 y el numero de argumentos totales), ejecuta acciones según el argumento que se le pasa, ejecuta al demonio cuando no esta en ejecución y reinicia todas las estructuras si no había ningún demonio ejecutándose.

Para apoyar la explicación de la implementación, se irán adjuntando screenshots del código, y comentaré lo mas relevante o lo que haya tenido problemas para codificar.

Inicio. Al inicio de la ejecución de Fausto, lo primero que nos encontramos son dos variables, obtenidas del ejercicio 1, para extraer el pid del demonio, principalmente en este archivo se han usado para pruebas y para comprobar y entender como funcionaba greo y cut. Si al hacer el grep de la linea_demonio, la variable esta vacía (por ello la comprobación del primer if), significa que no existe demonio y tenemos que reiniciar todas estructuras. Para ello, se comprueba mediante ifs que los diferentes archivos existen, y si existen los borra. Además, en el directorio, se hace una comprobación adicional para comprobar si tiene archivos dentro o no, así lo borramos con rmdir (si esta vacio) o con rm –rf si tiene archivos dentro.

Después, los echos posteriores y el mkdir, lo único que hacen es crear de nuevo los archivos.

Fausto.sh (~/Escritorio/DyASO_PED1_Osta_Supervia_Noelia/Trabajo

Abrir

```
#!/bin/bash

#Recibe órdenes creando los procesos y listas adecuadas

#Variables
linea_demonio=$(ps l | grep [D]emonio)
pid_demonio=$(echo $linea_demonio | cut -d " " -f3)

#Si el Demonio no está vivo lo crea
#Al leer/escribir en las listas hay que usar bloqueo para no co
if [[ -z $linea_demonio ]]
  then
  #No existe, por tanto, reinicia todas las estructuras
  if [[ -f 'procesos' ]]
    then rm 'procesos'
  fi
  if [[ -f 'procesos_servicio' ]]
    then rm 'procesos_servicio'
  fi
  if [[ -f 'procesos_periodicos' ]]
    then rm 'procesos_periodicos'
  fi
  if [[ -f 'Biblia.txt' ]]
    then rm 'Biblia.txt'
  fi
  if [[ -f 'San Pedro' ]]
    then rm 'San Pedro'
  fi
  if [[ -f 'Apocalipsis' ]]
    then rm 'Apocalipsis'
  fi
  if [[ -d 'Infierno' ]]
  then
    if [ "$(ls Infierno)" ]
    then
      rm -rf Infierno
      else
        rmdir Infierno
    fi
  fi

  echo > procesos
  echo > procesos_servicio
  echo > procesos_periodicos
  echo > Biblia.txt
  echo > 'San Pedro'
  mkdir Infierno
```

Inicialización demonio. Como se indica en la práctica, lo lanzamos con nohup, para que el proceso se ejecute en segundo plano, y redirigimos la salida a dev/null (vamos, que redirigimos la salida al "cubo de basura") y con la & indicamos que se ejecuta el script ./Demonio.sh en segundo plano. Además, creamos una entrada en la biblia. Se usa echo –e porque estoy usando \n para que me haga un salto de línea, y como va a ser la primera inicialización de la Biblia, hago un > y "sobreescribo" entre comillas porque esta vacío, el archivo Biblia.txt. Con esto hemos terminado el primer if de la comprobación acerca de si existia proceso demonio, por tanto, todo lo que se ejecute después se hará indiferentemente de que hubiera un demonio en ejecución o no.

```
  #Creamos el demonio
  nohup ./Demonio.sh > /dev/null &
  #Escribimos nacimiento en la biblia
  echo -e $(date +%T) '--------------- Génesis --------------\n'$(date +%T)' El demonio ha sido creado' > Biblia.txt
fi

case $1 in
```

Case. Comprobamos el argumento que se ha pasado a Fausto.sh al lanzar el comando, la variable $1 indica el primer argumento, por tanto, podemos saber si se le ha pasado el comando, run, run-service, etc. Dependiendo del comando entra en una opción u otra.

En la primera opción, lanzamos en segundo plano una instancia de bash con –c para indicarle que vamos a pasarle un argumento, (el cual sera $2, el segundo arigumento que le hayamos pasado a Fausto.sh), después, usamos un "cerrojo" para comprobar que nadie mas esta escribiendo en la lista procesos y escribimos el nacimiento en la Biblia tomando $! Para conseguir el pid del último proceso que se ha ejecutado. El comando run-service es idéntico cambiando que se escribe en la lista de procesos de servicio. Veo interersante comentar run-periodic, puesto que el argumento que se le pasa a bash es $3 (que es el comando que queremos ejecutar) y al escribir en procesos periódicos ponemos un 0 (puesto que todavía no ha comenzado su ejecución), $2 porque es el tiempo de ejecución, y $3.

List lo unico que hace es imprimir las tres listas por pantalla. He usado awk porque es el más usado en la práctica. Help lo unico que hace son varios echos para mostrar una "ayuda" por pantalla.

```
case $1 in
  #menos c para indicar que se le estan pasando argumentos como comandos
  run)
    bash -c "$2" &
    #Usamos $! para obtener el pid del último proceso creado
    flock 'San Pedro' echo $! "'$2'" >> procesos
    echo $(date +%T) El proceso $! "'$2'" ha nacido. >> Biblia.txt
  ;
  run-service)
    bash -c "$2" &
    flock 'San Pedro' echo $! "'$2'" >> procesos_servicio
    echo $(date +%T) El proceso $! "'$2'" ha nacido. >> Biblia.txt
  ;;
  run-periodic)
    bash -c "$3" &
    flock 'San Pedro' echo '0' $2 $! "'$3'" >> procesos_periodicos
    echo $(date +%T) El proceso $! "'$3'" ha nacido. >> Biblia.txt
  ;;
  list)
  awk {print} 'procesos'
  awk {print} 'procesos_servicio'
  awk {print} 'procesos_periodicos'
  ;;
 help)
  echo '-------------- Lista de comandos --------------'
  echo 'run             --> ./Fausto.sh run comando'
  echo 'run-service     --> ./Fausto.sh run-service comando'
  echo 'run-periodic    --> ./Fausto.sh run-periodic T comando'
  echo 'list            --> ./Fausto.sh list'
  echo 'help            --> ./Fausto.sh help'
  echo 'stop            --> ./Fausto.sh stop PID'
  echo 'end             --> ./Fausto.sh end'
  ;;
```

Stop. Para cada archivo, comprobamos con awk y un if si el argumento del pid que le hemos pasado a Fausto ($2) es el mismo que algún pid en algún archivo. Este pid en procesos y procesos_servicio se almacena en la columna 1, y en procesos servicio en la columna 3, por eso la distinción en awk if($3 ...)

Después, se comprueba que alguna de esas variables no está vacía, y su no lo esta, creamos un nuevo archivo en Infierno con el nombre del pid.

Apocalipsis. Sólo crea el fichero apocalipsis para que lo detecte demonio y se encarge del resto.

Por último, el default del case, consiste en que si ninguno de los comandos ha sido encontrado, mostrara un mensaje como que el comando no esta reconocido.

```
  ;;
  stop)
  pid_procesos=$(awk '{if ($1=="'"$2"'") print $1}' procesos)
  pid_procesos_servicio=$(awk '{if ($1=="'"$2"'") print $1}' procesos_servicio)
  #El archivo procesos periodicos contiene la informacion del PID en la columnna 3. Por tanto con awk hacemos un if
para comprobar que la primera coluna es la misma que el pid que nos pasan en la variable 2
  pid_procesos_periodicos=$(awk '{if ($3=="'"$2"'") print $3}' procesos_periodicos)
  if [[ $pid_procesos || $pid_procesos_servicio || $pid_procesos_periodicos ]]
  then
    > Infierno/"$2"
    else
    echo 'Introduzca un PID válido'
  fi

  ;;
  end)
  > Apocalipsis
  ;;
  *)
    echo Error, orden $1 no reconocida, consulte las órdenes disponibles con ./Fausto.sh help
  ;;
esac
```

Demonio.sh Se declaran las mismas variables que en Fausto y además la función que proporcionó el ED para matar al proceso y a todos sus descendientes.

Demonio.sh (~/Escritorio/DyASO_PED1_Osta_Supervia_Noelia/Trabajo1) - gedit

Abrir

```
#!/bin/bash

#Variables
linea_demonio=$(ps l | grep [D]emonio)
pid_demonio=$(echo $linea_demonio | cut -d " " -f3)
#Funcion
function matar_proceso_y_descendientes(){
#proporcionada por el ED
    pid_padre=$1
    pids=$(pstree -p $pid_padre | grep -o -E '[0-9]+')
    pids=$(echo $pids | tr '\n' ' ')
    kill -SIGTERM $pids
}
```

Inicio. Lo primero que se hace en Demonio, es comprobar que no existe ningún archivo Apocalipsis, mientras este archivo no exista, el recorrerá todas las listas ejecutando las acciones que se han requerido en el enunciado de la práctica, y cuando exista apocalipsis, procedera a borrar todas las listas, finalizar todos los procesos, y a finalizarse el mismo.

#Lista procesos. Se ejecuta un bucle for por cada pid que se encuentre en la lista de procesos. Lo primero que haremos es comprobar si Infierno esta vacio, si no lo esta, comprobaremos entre los achivos de Infierno que existe el mismo haciendo un grep con el pid que hemos obtenido en el bucle for. Si en el infierno existe el mismo archivo, lo primero que hago es guardarme el comando del proceso haciendo un awk para sacar la linea del proceso (por eso el print 0), y luego con cut, obtengo

simplemente el comando. También esta explicado en los comentarios del código. Luego borramos de la lista de procesos la línea que contiene el pid del proceso, borramos en infierno con rm el archivo del pid, escribimos en la Biblia su fallecimiento, y usamos la función del ED para matar el proceso y todos sus descendientes (recordar que con $! Estamos cogiendo el pid de bash, por tanto, también tendremos que buscar a todos los hijos que tengan ese pid como padre y eliminarlos). Si no esta en el infierno, comprobamos si esta en ejecución y si lo esyá, no hacemos nada, si no lo estuviera, hacemos los pasos anteriores como si mataramos al proceso pero sin llamar a la función recursiva puesto que el proceso no está en ejecución.

```
  until [ -f "Apocalipsis" ]
  do
    sleep 1
#   -Lee las listas y revive los procesos cuando sea necaario dejando entradas en la biblia
  #Lista procesos
  for pid in $(awk '{print $1}' procesos)
    do
    if [[ "$(ls Infierno)" ]]
      then
        #Comprobamos si en infierno esta el mismo
        existe=$(ls Infierno | grep "$pid")
        if [ "$existe" ]
          then
          #Encuentra una coincidencia, procedemos a borrar todo su árbol de procesos, su entrada en la lista, el
fichero correspondiente en Infierno, ademas, creamos entrada en la biblia.
          #Primero, guardamos el argumento para poder almacenarlo en la biblia
          # Con awk extraemos la fila entera que contiene el pid, luego lo cortarmos y con -f2- le indicamos que
queremos desde f2 hasta el final del archivo lo cual nos da el comando
          command_line=$(awk '{if ($1=="'"$pid"'") print $0}' procesos | cut -d ' ' -f2-)
          flock 'San Pedro' sed -i "/$pid/d" procesos
          rm Infierno/$pid
          echo $(date +%T) El proceso $pid $command_line ha terminado. >> Biblia.txt
          matar_proceso_y_descendientes "$pid"
        fi
    else
      #No se encuentra en el infierno, por tanto, comprobamos si sigue en ejecucion
      echo "No se encuentra en el infierno"
      ejecucion=$(ps | grep "$pid")
      if [ "$ejecucion" ]
        then
          echo "esta en ejecucion, no hacemos nada"
          else
          #No esta en ejecucion, lo borramos y anotamos su entrada en la biblia
          command=$(awk '{if ($1=="'"$pid"'") print $0}' procesos | cut -d ' ' -f2-)
          comando_sin_comillas=$(echo "$command" | tr -d "'")
          flock 'San Pedro' sed -i "/$pid/d" procesos
          echo $(date +%T) El proceso $pid $command ha terminado. >> Biblia.txt
      fi
    fi
  done
```

Lista procesos servicio. Bastante similar a la anterior, salvo que, si el proceso no está en ejecución, lo resucitamos, es decir, lanzamos un nuevo proceso con el mismo comando que el proceso que ya no está ejecutándose, además, borramos la entrada anterior de la lista de procesos con sed buscando la linea del pid (y con –i para indicar que sobreescribimos el archivo). Siempre que se escribe en las listas, se usa flock para asegurarnos que nadie mas esta escribiendo.

```
  #Lista procesos_servicio
  for pid in $(awk '{print $1}' procesos_servicio)
    do
    if [[ "$(ls Infierno)" ]]
      then
        #Comprobamos si en infierno esta el mismo
        existe=$(ls Infierno | grep "$pid")
        if [ "$existe" ]
          then
          #Encuentra una coincidencia, procedemos a borrar todo su árbol de procesos, su entrada en la lista, el
fichero correspondiente en Infierno, ademas, creamos entrada en la biblia.
          #Primero, guardamos el argumento para poder almacenarlo en la biblia
          # Con awk extraemos la fila entera que contiene el pid, luego lo cortarmos y con -f2- le indicamos que
queremos desde f2 hasta el final del archivo lo cual nos da el comando
          command_line=$(awk '{if ($1=="'"$pid"'") print $0}' procesos_servicio | cut -d ' ' -f2-)
          flock 'San Pedro' sed -i "/$pid/d" procesos_servicio
          rm Infierno/$pid
          echo $(date +%T) El proceso $pid $command_line ha terminado. >> Biblia.txt
          matar_proceso_y_descendientes "$pid"
        fi
    else
      #No se encuentra en el infierno, por tanto, comprobamos si sigue en ejecucion
      ejecucion=$(ps | grep "$pid")
      if [ "$ejecucion" ]
        then
          echo "esta en ejecucion, no hacemos nada"
          else
          #No esta en ejecucion, resucitamos al proceso y lo agregamos a la biblia
          command=$(awk '{if ($1=="'"$pid"'") print $0}' procesos_servicio | cut -d ' ' -f2-)
          comando_sin_comillas=$(echo "$command" | tr -d "'")
          #Borramos el anterior de la lista de procesos y lo creamos
          flock 'San Pedro' sed -i "/$pid/d" procesos_servicio
          bash -c "$comando_sin_comillas" &
          flock 'San Pedro' echo $! $command >> procesos_servicio
          #Ponemos el nuevo proceso creado en la lista de procesos
          echo $(date +%T) El proceso $pid $command resucita con pid $! >> Biblia.txt
      fi
    fi
  done
```

Lista procesos periódicos. Similar a las anteriores, exceptuando que como los procesos son periódicos, tendremos que comprobar los tiempos (obtenidos tambien con awk de las columnas 1 y 2 de la lista de procesos_periodicos). Aqui no supe interpretar muy bien el enunciado, y entendí que lo **primero** que se hacia era actualizar el tiempo y luego se comprobaba que el tiempo era mayor que el tiempo total. Por tanto eso es lo que se hace, se actualiza el tiempo de la lista de procesos con sed reemplazando el tiempo por la variable nuevo tiempo, y después se comprueba que el proceso está en ejecución. Si el proceso está ejecutándose, no hacemos nada, pero si no lo está, comprobamos que el tiempo (que se acaba de incrementar) es mayor que el tiempo total de ejecución [-eq nos lo comprueba] y si lo es, reencarnamos el proceso volviendo a lanzar una instancia de bash con el parámetro en segundo plano, actualizando la Biblia, y cambiando las listas.

```
      #Lista procesos_periodicos
  for pid in $(awk '{print $3}' procesos_periodicos)
    do
    #Primero, guardamos el argumento para poder almacenarlo en la biblia
    # Con awk extraemos la fila entera que contiene el pid, luego lo cortarmos y con -f2- le indicamos que quere
desde f2 hasta el final del archivo lo cual nos da el comando
    command=$(awk '{if ($3=="'"$pid"'") print $0}' procesos_periodicos | cut -d ' ' -f4-)
    comando_sin_comillas=$(echo "$command" | tr -d "'")
    if [[ "$(ls Infierno)" ]]
      then
        #Comprobamos si en infierno esta el mismo
        existe=$(ls Infierno | grep "$pid")
        if [ "$existe" ]
          then
          #Encuentra una coincidencia, procedemos a borrar todo su árbol de procesos, su entrada en la lista, e
fichero correspondiente en Infierno, ademas, creamos entrada en la biblia.
          flock 'San Pedro' sed -i "/$pid/d" procesos_periodicos
          rm Infierno/$pid
          echo $(date +%T) El proceso $pid $command ha terminado. >> Biblia.txt
          matar_proceso_y_descendientes "$pid"
        fi
    else
      #No se encuentra en el infierno, por tanto, incrementamos el contador de tiempo en todos
      tiempo=$(awk '{if ($3=="'"$pid"'") print $1}' procesos_periodicos )
      tiempo_total=$(awk '{if ($3=="'"$pid"'") print $2}' procesos_periodicos )
      nuevo_tiempo=$((tiempo+1))
      #Explicacion detallada, usamos -i para que sobreescriba el mismo fichero , primero buscamos la linea del p
indicando que es el inicio de la linea con ^, todo ello del archivo de procesos periodicos
      flock 'San Pedro' sed -i "/$pid/ s/^$tiempo /$nuevo_tiempo /g" procesos_periodicos
      ejecucion=$(ps | grep "$pid")
      if [ "$ejecucion" ]
        then
          echo "esta en ejecucion, no hacemos nada, solo el incrementar contador que ya esta incrementado"
         else
          #No esta en ejecucion
          echo "No esta en ejecucion, miramos el tiempo y lo volvemos a lanzar"
          if [[ $nuevo_tiempo -ge $tiempo_total ]]
          then
            echo "Resucitar"
            #Borramos el anterior de la lista de procesos y lo creamos
            flock 'San Pedro' sed -i "/$pid/d" procesos_periodicos
            bash -c "$comando_sin_comillas" &
            flock 'San Pedro' echo '0' $tiempo_total $! $command >> procesos_periodicos
            echo $(date +%T) El proceso $pid $command se ha reencarnado en el pid $! >> Biblia.txt
          fi
      fi
    fi
```

Apocalipsis. Conforme empieza el apocalipsis, se anota una entrada en la biblia. Aqui también mediante awk obtenemos todos los pid de las listas de procesos, procesos_servicio y procesos_periodicos. Por cada pid que exista, matamos todo el arbol de procesos con la función del ED, ademas, borramos con flock y sed su entrada de la lista de procesos buscando la línea del pid correspondiente, y anotamos la entrada en la Biblia. Esto es igual para las 3 listas, con la excepción de que cada sed se hace sobre la lista correspondiente, y que en la lista de procesos periodicos el pid se encuentra en la columna 3 al hacer el awk.

Una vez que hemos eliminado todos los procesos que quedaban en las listas, borramos todos los archivos con rm. Y por último, con el pid que teniamos del proceso demonio, el mismo es el encargado de terminar su ejecución.

```
#Apocalipsis: termino todos los procesos y limpio todo dejando sólo Fausto, el Demonio y la Biblia
echo $(date +%T) '--------------- Apocalipsis -------------' >> Biblia.txt
  #Lista procesos
  for pid in $(awk '{print $1}' procesos)
    do
      #Para cada pid que exista, matamos el todo el arbol de procesos
      flock 'San Pedro' sed -i "/$pid/d" procesos
      echo $(date +%T) El proceso $pid ha terminado. >> Biblia.txt
      matar_proceso_y_descendientes "$pid"
  done

  #Lista procesos servicio
  for pid in $(awk '{print $1}' procesos_servicio)
    do
      #Para cada pid que exista, matamos el todo el arbol de procesos
      flock 'San Pedro' sed -i "/$pid/d" procesos_servicio
      echo $(date +%T) El proceso $pid ha terminado. >> Biblia.txt
      matar_proceso_y_descendientes "$pid"
  done

      #Lista procesos periodicos
  for pid in $(awk '{print $3}' procesos_periodicos)
    do
      #Para cada pid que exista, matamos el todo el arbol de procesos
      flock 'San Pedro' sed -i "/$pid/d" procesos_periodicos
      echo $(date +%T) El proceso $pid ha terminado. >> Biblia.txt
      matar_proceso_y_descendientes "$pid"
  done
```

o Ejecución de ejemplo

En este apartado se discutirá la traza de ejecución del programa en base a los datos de ejemplo aportados en el guión, demostrando así que el algoritmo funciona correctamente.

Paso 1. Se comprueba que el demonio se crea correctamente. Como se puede comprobar esta desacoplado de bash.

```
sistemas@DyASO:~/Documentos/DyASO_PED1_Osta_Supervia_Noelia$ ./Ejercicio1.sh
nohup: se redirige la salida de error a la salida estándar

************************************************************
1) Debería de haberse creado el proceso Demonio
la salida esperada es algo así
systemd---systemd---Demonio.sh---sleep
donde Demonio.sh no debe ser hijo de bash sino de systemd o similar
************************************************************

systemd---lightdm---lightdm---upstart---Demonio.sh---sleep

************************************************************
```

Paso 2. Creación de procesos. Aparecen los procesos creados con su pid en las listas, y se muestran las listas correctamente en fausto list.

```
**********************************************************
2) Lanzo algunos comandos y compruebo que se han creado
Debería de haber un proceso normal
'sleep 10; echo hola > test1.txt'
Un proceso servicio 'yes > /dev/null'
y dos periódicos, el normal y el lento
Comparamos los procesos teóricamente lanzados y los que
realmente existen
**********************************************************

Procesos lanzados según Fausto:
./Fausto.sh list

8978 'sleep 10; echo hola >> test1.txt'

9003 'yes > /dev/null'

0 5 9015 'echo hola_periodico >> test2.txt'
0 5 9026 'echo hola_periodico_lento >> test3.txt; sleep 20'

Procesos existentes:
ps -l
F S   UID    PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000   8951  8944  0  80   0 -  2048 wait   pts/4    00:00:00 bash
0 S  1000   8961  8951  0  80   0 -  1676 wait   pts/4    00:00:00 Ejercicio1.sh
0 S  1000   8974  1749  0  80   0 -  1682 wait   pts/4    00:00:00 Demonio.sh
0 S  1000   8978  1749  0  80   0 -  1672 wait   pts/4    00:00:00 bash
0 S  1000   8983  8978  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 S  1000   8991  8974  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 S  1000   9003  1749  0  80   0 -  1672 wait   pts/4    00:00:00 bash
0 R  1000   9006  9003  5  80   0 -  1375 -      pts/4    00:00:00 yes
0 S  1000   9026  1749  0  80   0 -  1672 wait   pts/4    00:00:00 bash
0 S  1000   9029  9026  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 R  1000   9041  8961  0  80   0 -  2191 -      pts/4    00:00:00 ps

**********************************************************
```

Paso 3. Se elimina manualmente el proceso que en la imagen anterior tenia el pid 9003, y como se puede comprobar, Demonio lo detecta en su bucle y mientras lee las listas reinicia el proceso, por lo tanto el proceso pasa a tener el pid 9062. Se ha resaltado para mayor comodidad.

```
**********************************************************
3) Elimino manualmente el proceso yes sin avisar a Fausto.
El Demonio debería detectarlo y reiniciar el proceso:
**********************************************************

pkill yes

bash: línea 1:  9006 Terminado              yes > /dev/null
./Fausto.sh list

8978 'sleep 10; echo hola >> test1.txt'

9062 'yes > /dev/null'

2 5 9015 'echo hola_periodico >> test2.txt'
2 5 9026 'echo hola_periodico_lento >> test3.txt; sleep 20'

**********************************************************
```

Paso 4. Se elimina el proceso, Fausto habrá creado una carpeta en el infierno con el nombre de archivo del pid del proceso, y demonio, cuando lo detecte, matará al proceso padre y a todos sus hijos.

Como se puede comprobar en el listado, no hay ningún proceso con el pid 9062. Aqui también podemos ver, que el proceso que al iniciarse tenia el pid 9015, se ha reencarnado, es decir, durante la ejecución, su tiempo habrá superado al tiempo total, el Demonio lo habrá detectado y creado un nuevo proceso con pid 9283. Es decir, el pid 9015 se ha reencarnado en el pid 9283.

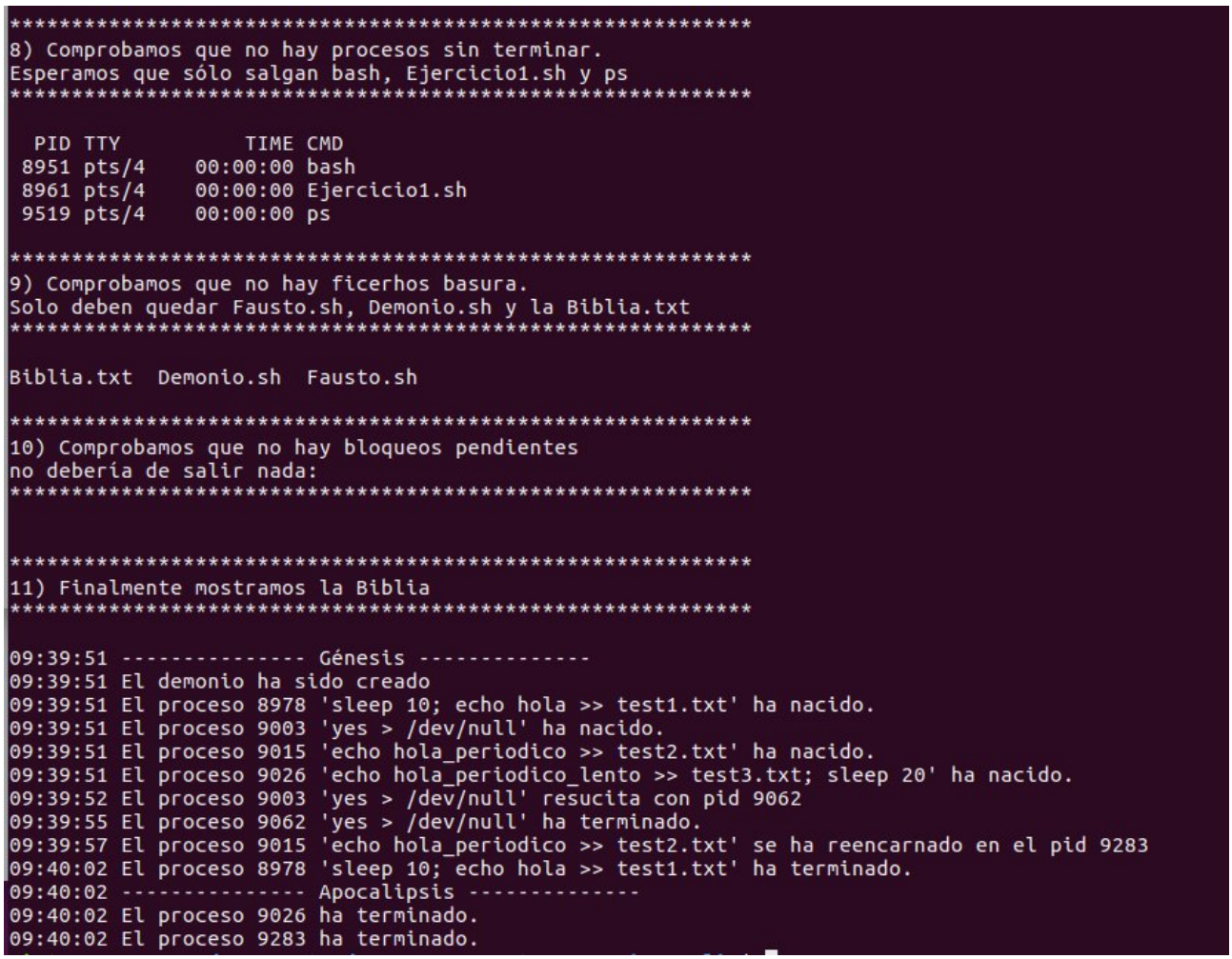
```
**********************************************************
4) Elimino el proceso usando Fausto.
El Demonio NO debe reiniciar el proceso:
**********************************************************

./Fausto.sh stop 9062
./Fausto.sh list

8978 'sleep 10; echo hola >> test1.txt'


5 5 9026 'echo hola_periodico_lento >> test3.txt; sleep 20'
0 5 9283 'echo hola_periodico >> test2.txt'
ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  8951  8944  0  80   0 -  2048 wait   pts/4    00:00:00 bash
0 S  1000  8961  8951  0  80   0 -  1676 wait   pts/4    00:00:00 Ejercicio1.sh
0 S  1000  8974  1749  0  80   0 -  1766 wait   pts/4    00:00:00 Demonio.sh
0 S  1000  8978  1749  0  80   0 -  1672 wait   pts/4    00:00:00 bash
0 S  1000  8983  8978  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 S  1000  9026  1749  0  80   0 -  1672 wait   pts/4    00:00:00 bash
0 S  1000  9029  9026  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 S  1000  9301  8974  0  80   0 -  1376 hrtime pts/4    00:00:00 sleep
0 R  1000  9312  8961  0  80   0 -  2191 -      pts/4    00:00:00 ps
```

Paso 5,6,7. Se comprueba que el default del case de Fausto está implementado correctamente y por tanto funciona que si no detecta comando muestra el mensaje de error, también, nos muestra la ayuda correctamente con el comando help. Además, podemos observar que hasta el momento, han pasado poco mas de 10 segundos, y le ha dado tiempo a ejecutar 1 hola, dos hola periodico, y un hola periodico lento.

Pasos 8,9,10, 11. Comprobamos que el Apocalipsis se ha llevado a cabo correctamente, y ha recorrido todas listas eliminando los procesos, también ha borrado correctamente todos los ficheros. Como los bloqueos se hacen únicamente en escritura es normal no tener nigún bloqueo pendiente. Además,en la biblia, comprobamos los pasos que se han ido indicando anteriormente:

1. Se crean los procesos 8978, 9003, 9015 y 9026. Como se mato al proceso 9003, vemos que resucita con pid 9062. Se elimina 3s después porque con fausto stop lo mandamos al Infierno y le estamos dando al demonio la orden de eliminarlo.

2. A los 6 segundos, en el 09:39:57 vemos que el proceso 9015 se reencarna en el proceso 9283 porque su tiempo sería mayor que el tiempo total, por lo tanto, el funcionamiento es correcto (realmente debería ser a los 5s pero Demonio no tarda 1 segundo justo en efectuar su recorrido de listas, así que le damos un pelín mas de margen). Sin embargo, al proceso 9026 no le da tiempo a reencarnarse porque sucede el apocalipsis. Observamos también que casi a los 10 segundos (realmente 11), el proceso 8978 termina, y el resto de procesos terminan con el apocalipsis. En el ejemplo que habéis proporcionado en el enunciado de la práctica, se observa que tenéis la terminación del proceso 7854 duplicada. Por lo tanto, yo tengo una entrada menos en el apocalipsis pero lo considero correcto también.



o Otros apartados

Comento los problemas surgidos ya que han sido bastantes. En algún momento, tuve un gran problema con la MV, con imposibilidad de recuperar los archivos y volver a empezar, lo cual fue muy tedioso y complico muchísimo la elaboración de la PEC. También tuve muchísimos problemas para compartir carpeta. Creo que hubiera sido bastante mejor, instalar una partición de Linux en el SO y yo personalmente me hubiera evitado muchos problemas.

· Bibliografía.

https://geekland.eu/uso-del-comando-cut-en-linux-y-unix-con-ejemplos/

Explicación nohup

https://atareao.es/software/utilidades/nohup/#:~:text=Lo%20primero%20es%20indicarte%20que,aunque%20tu%20salgas%20del%20terminal.

Cut, awk, sed, grep

https://atareao.es/tutorial/terminal/filtros-awk-grep-sed-y-cut/

En general, todo lo que he necesitado, estaba en la página de atareao o stack overflow. En todos los casos, se ha adaptado el código a lo requerido y excepto la función para matar los procesos proporcionada por el ED y la de sacar el pid del Demonio. El resto ha sido trabajo de búsqueda, codificación, ensayo, error y muchos muchos echos (para hacer una especie de debug).