

# Trabajo de Diseño y Administración de Sistemas Operativos

**ALUMNO: JORGE MARTÍNEZ PAZOS**

**DNI: 39510046-W**

**CENTRO ASOCIADO: VIGO (PONTEVEDRA)**

**TELÉFONO DE CONTACTO: 630005088**

**EMAIL: JMARTINEZ5741@ALUMNO.UNED.ES**

# Primera PED

## Introducción

En este trabajo se ha implementado un gestor de procesos que consta de dos componentes principales: **Fausto** y **Demonio**. Fausto actúa como una interfaz de línea de comandos que permite al usuario gestionar procesos, mientras que Demonio es el encargado de supervisar y gestionar dichos procesos en segundo plano.

Se han desarrollado las funcionalidades de creación de procesos normales, servicios y procesos periódicos, asegurando una correcta sincronización entre Fausto y Demonio mediante el uso de bloqueos. Además, se ha implementado el manejo del *Apocalipsis*, que limpia todos los recursos al final de la ejecución, y se registran todas las acciones realizadas en la Biblia, un archivo de log detallado del sistema.

## Implementación

Este trabajo tiene como objetivo implementar un gestor de procesos en Bash que permita explorar conceptos fundamentales de los sistemas operativos, como la creación, monitorización y terminación de procesos, así como la sincronización y la comunicación entre scripts concurrentes.

### 1. Estructura del Sistema

El sistema se compone de los siguientes elementos:

- **Fausto.sh**: Interfaz de usuario que, mediante comandos del usuario, permite ejecutar, gestionar y detener procesos. También puede manejar procesos normales, servicios (que continúan ejecutándose en segundo plano independientemente del terminal) y procesos periódicos (que se reencarnan a intervalos definidos). Interactúa con los archivos de listas de procesos y comunica con **Demonio.sh** para realizar las tareas de gestión.
- **Demonio.sh**: Proceso en segundo plano encargado de monitorizar y reiniciar los procesos ejecutados. Se asegura de que los procesos se mantengan activos y se reencarnen si es necesario. También gestiona el ciclo del *Apocalipsis*, donde se eliminan todos los procesos y archivos asociados al finalizar la ejecución.

### 2. Funcionalidades Implementadas

#### 1. Gestión de Procesos:

- a. **Ejecutar procesos:** Fausto permite ejecutar procesos de forma normal, como servicios (que siguen ejecutándose en segundo plano) y como procesos periódicos (que se ejecutan a intervalos de tiempo definidos).
  - b. **Reencarnación de procesos:** Los procesos periódicos se reencarnan automáticamente cuando alcanzan su tiempo de ejecución, y los servicios se reinician si se detienen inesperadamente.
2. **Comunicación entre Fausto y Demonio:**
- a. **Bloqueo de acceso a las listas:** Se implementó un sistema de bloqueo para evitar que Fausto y Demonio accedan a las listas de procesos simultáneamente, lo que podría provocar inconsistencias.
  - b. **Monitorización y reinicio de procesos:** Demonio monitoriza constantemente los procesos y reinicia aquellos que han finalizado o que deben reencarnarse después de alcanzar el período establecido.
3. **Apocalipsis:**
- a. Al finalizar la ejecución, el sistema ejecuta el *Apocalipsis*, un proceso que limpia todos los recursos utilizados (archivos de procesos, directorios y demás datos temporales).
4. **Registro de Acciones en la Biblia:**
- a. Todas las acciones realizadas por Fausto y Demonio se registran en un archivo llamado **Biblia.txt**. Esto incluye la creación de procesos, el reinicio de los mismos, el manejo de errores y el ciclo del *Apocalipsis*.

### 3. Resultados Obtenidos

Durante las pruebas del sistema, los siguientes resultados fueron observados:

1. **Gestión de Procesos:**
- a. Los procesos normales se ejecutan correctamente y se terminan al completarse.
  - b. Los procesos de servicio se mantienen en ejecución en segundo plano incluso si el terminal se cierra o si Fausto se detiene.
  - c. Los procesos periódicos se ejecutan de acuerdo con el intervalo definido, se reencarnan correctamente después de alcanzar su período, y se escriben en los archivos correspondientes (por ejemplo, `test2.txt`, `test3.txt`).
2. **Sincronización entre Fausto y Demonio:**
- a. Los bloqueos entre Fausto y Demonio funcionan correctamente, evitando la manipulación simultánea de las listas de procesos.
  - b. No se produjeron errores relacionados con el acceso simultáneo a los archivos de procesos.
3. **Apocalipsis:**

- a. Al finalizar la ejecución, el *Apocalipsis* borra correctamente los archivos y directorios utilizados para almacenar los procesos.
- b. Todos los procesos se eliminan correctamente al término de la ejecución, y la Biblia refleja el proceso de limpieza.

#### 4. Errores:

- a. Durante las pruebas iniciales, se presentaron algunos errores relacionados con la eliminación de procesos y la actualización de las listas. Sin embargo, después de revisar el manejo de los PIDs y corregir algunos errores en las funciones de reencarnación, el sistema comenzó a funcionar de manera estable.

## Conclusión:

El sistema ha sido implementado con éxito y cumple con los objetivos establecidos. La comunicación entre Fausto y Demonio es fluida, los procesos periódicos se gestionan correctamente, y el ciclo del *Apocalipsis* garantiza que los recursos se limpien al finalizar la ejecución. Las pruebas indican que el sistema está funcionando según lo esperado, con la excepción de algunas pequeñas optimizaciones que se realizarán en futuras iteraciones.

## Ejecución de ejemplo

Comando ejecutado:

```
./Fausto.sh run 'sleep 10; echo hola >> test1.txt'
```

Explicación:

Se verifica que el sistema inicia correctamente. Fausto detecta que Demonio no está activo y lo lanza, registrándolo en Biblia.

Captura:

```
18:53:47 [Fausto] -----Génesis-----
18:53:47 [Fausto] El demonio ha sido creado
18:53:47 [Fausto] El proceso 1402 'sleep 10; echo hola >> t
nació.
```

Análisis:

La salida muestra que el demonio se creó correctamente, cumpliendo lo esperado.

\*\*\*\*\*

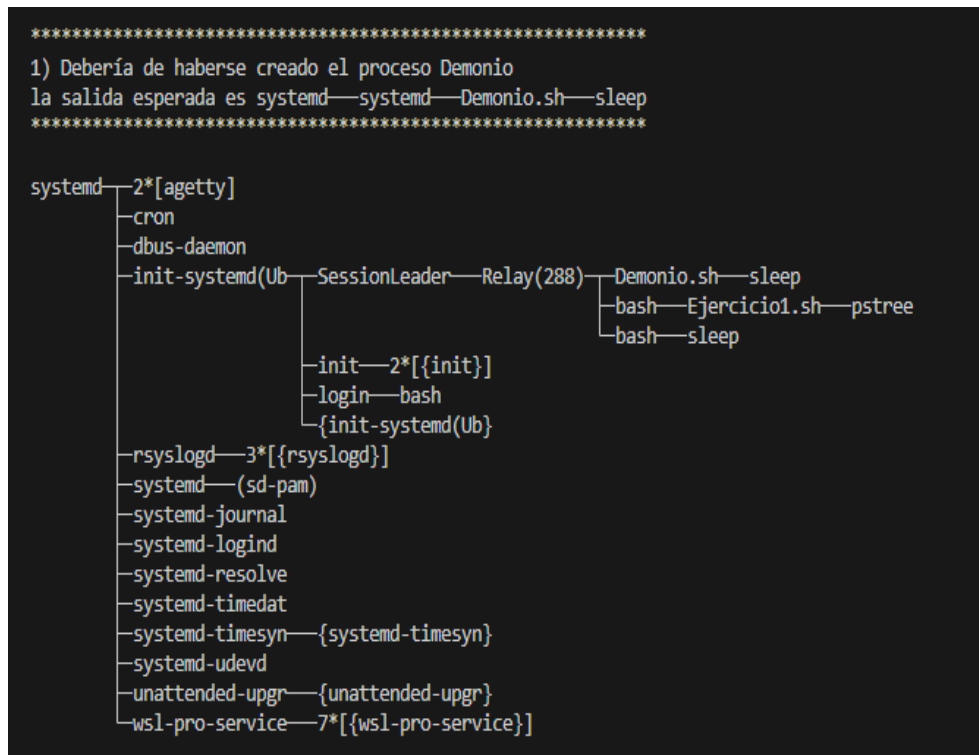
Comando ejecutado:

```
pstree -s $(ps l | grep [D]emonio | cut -d " " -f5)
```

Explicación:

Comprobamos que efectivamente se ha lanzado el proceso demonio y que ha sido adoptado por systemd.

Captura:



Análisis:

La salida muestra todo el árbol de procesos, fui incapaz de conseguir que solo se creasen o mostrase el árbol de 4 procesos. En todo caso demuestra la existencia del Demonio y su correcta creación.

\*\*\*\*\*

Comando ejecutado:

```
./Fausto.sh run 'sleep 10; echo hola >> test1.txt'
```

```
./Fausto.sh run-service 'yes > /dev/null'
```

```
./Fausto.sh run-periodic 5 'echo hola_periodico >> test2.txt'
```

```
./Fausto.sh run-periodic 5 'echo hola_periodico_lento >> test3.txt; sleep 20'
```

Explicación:

Ejecutamos algunos comandos para ir añadiendo a sus respectivas listas.

Captura:

```
18:53:47 [Fausto] El demonio ha sido creado
18:53:47 [Fausto] El proceso 1402 'sleep 10; echo hola >> test1.txt' ha
nacido.
18:53:47 [Fausto] El proceso 1415 'yes > /dev/null' ha nacido.
18:53:48 [Fausto] El proceso '1422' 'echo hola_periodico >> test2.txt'
ha nacido para ejecutarse periódicamente cada 5 segundos.
18:53:49 [Fausto] El proceso '1437' 'echo hola_periodico_lento >> test3.
txt; sleep 20' ha nacido para ejecutarse periódicamente cada 5 segundos.
```

Análisis:

La salida muestra cómo se han creado y registrado los procesos en la Biblia, indicando el tipo y su pid.

\*\*\*\*\*

Comando ejecutado:

```
./Fausto.sh list
```

```
ps -l
```

Explicación:

Pedimos que se nos muestre por pantalla los procesos registrados.

Captura:

```

*****
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
***** Procesos *****
437 'sleep 10; echo hola >> test1.txt'
***** Procesos_Servicio *****
450 'yes > /dev/null'
***** Procesos_Periodicos *****
0 5 457 'echo hola_periodico >> test2.txt'
0 5 472 '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
4 S   1000      288       287  0   80    0     - 1577  do_wai pts/0        00:00:00 bash
0 S   1000      422       288  0   80    0     - 1188  do_wai pts/0        00:00:00 Ejercicio1.sh
0 S   1000      435       287  0   80    0     - 1221  do_wai pts/0        00:00:00 Demonio.sh
0 S   1000      437       287  0   80    0     - 1188  do_wai pts/0        00:00:00 bash
0 S   1000      440       437  0   80    0     - 781   hrtime pts/0        00:00:00 sleep
0 S   1000      450       287  0   80    0     - 1188  do_wai pts/0        00:00:00 bash
0 R   1000      453       450 99   80    0     - 781   -      pts/0        00:00:02 yes
0 S   1000      472       287  0   80    0     - 781   hrtime pts/0        00:00:00 sleep
0 S   1000      497       435  0   80    0     - 782   hrtime pts/0        00:00:00 sleep
0 R   1000      502       422  0   80    0     - 2078  -      pts/0        00:00:00 ps

```

Análisis:

La salida muestra de manera correcta los procesos presentes ahora mismo en el sistema.

\*\*\*\*\*

Comando ejecutado:

pskill yes

./Fausto.sh list

Explicación:

Se elimina manualmente el proceso de nombre “yes”, debe reiniciarse automáticamente, a continuación, pedimos que se nos muestre por pantalla los procesos registrados.

Captura:

```

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

pkill yes

./Fausto.sh list
***** Procesos *****
437 'sleep 10; echo hola >> test1.txt'
***** Procesos_Servicio *****
450 'yes > /dev/null'
***** Procesos_Periodicos *****
1 5 457 'echo hola_periodico >> test2.txt'
1 5 472 'echo hola_periodico_lento >> test3.txt; sleep 20'

```

Análisis:

La salida devuelve los procesos concurrentes, enseñando cómo se ha eliminado con pkill y aun así el proceso ha sido resucitado y sigue funcionando.

\*\*\*\*\*

Comando ejecutado:

```
pid_yes=$(./Fausto.sh list | grep [y]es | cut -d " " -f1)
```

```
./Fausto.sh stop $pid_yes
```

```
./Fausto.sh list
```

Explicación:

Se elimina un proceso mediante Fausto, de modo que este no debe resucitar, primero se busca su pid, después se para el proceso y por último se llama a la lista de procesos concurrentes.

Captura:

```

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

./Fausto.sh stop 530
530 marcado para eliminación
./Fausto.sh list
***** Procesos *****
437 'sleep 10; echo hola >> test1.txt'
***** Procesos_Servicio *****
***** Procesos_Periodicos *****
4 5 457 'echo hola_periodico >> test2.txt'
4 5 472 'echo hola_periodico_lento >> test3.txt; sleep 20'

```



Análisis:

La salida muestra como la función stop detiene el proceso y lo elimina, sin llegar a resucitarlo.

\*\*\*\*\*

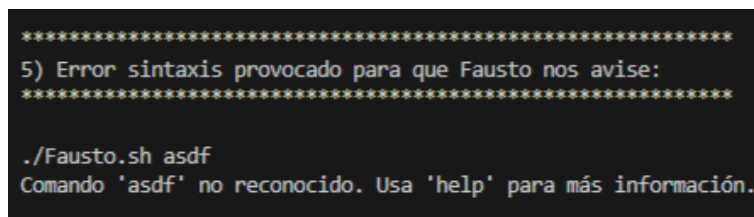
Comando ejecutado:

`./Fausto.sh asdf`

Explicación:

Se pasa un comando erróneo con el objetivo de comprobar que, efectivamente, se detectan errores en los comandos y no se acepta cualquier cosa que se introduzca.

Captura:

A screenshot of a terminal window with a dark background and light-colored text. The text shows a prompt followed by the command './Fausto.sh asdf'. Below the command, an error message is displayed: 'Comando 'asdf' no reconocido. Usa 'help' para más información.' The message is preceded by a line number '5)' and the text 'Error sintaxis provocado para que Fausto nos avise:'. The entire output is flanked by lines of asterisks.

```
*****
5) Error sintaxis provocado para que Fausto nos avise:
*****

./Fausto.sh asdf
Comando 'asdf' no reconocido. Usa 'help' para más información.
```

Análisis:

La salida muestra cómo, de manera acertada, se detecta el comando que no coincide con ninguno de los esperados y salta un error.

\*\*\*\*\*

Comando ejecutado:

`./Fausto.sh help`

Explicación:

Se lanza el comando help para que devuelva la lista de posibles combinaciones de comandos que acepta el programa Fausto.

Captura:

```

*****
6) Siguiendo la sugerencia anterior verifamos la ayuda:
*****

./Fausto.sh help
Uso: ./Fausto.sh [comando]

**** Comandos disponibles ****

run comando           Ejecuta un proceso normal
run-service comando   Ejecuta un proceso como servicio
run-periodic T comando Ejecuta un comando periódicamente cada T segundos
list                  Lista todos los procesos
help                  Muestra esta ayuda
stop PID              Detiene un proceso por su PID
end                   Finaliza el demonio y limpia recursos

```

## Análisis:

La salida devuelve efectivamente la lista de posibles comandos reconocidos por el programa.

\*\*\*\*\*

## Comando ejecutado:

./Fausto.sh end

cat test1.txt test2.txt test3.txt

rm test1.txt test2.txt test3.txt

## Explicación:

Se lanza el comando end, para iniciar el proceso de Apocalipsis y que termine tanto Fausto como el demonio, y se eliminan todos los archivos necesarios. A continuación se pide que se muestre por pantalla los archivos de texto test1,2 y 3 para observar cuántas veces se han llevado a cabo cada uno de los procesos.

## Captura:

```

*****
7) Terminamos la ejecución y vemos los mensajes enviados
por los procesos lanzados en los ficeheros test 1, 2 y 3
*****

hola
hola_periodico
hola_periodico
hola_periodico_lento

```

### Análisis:

La salida muestra como el proceso hola no se repitió, el proceso hola\_periódico se repitió 2 veces y el proceso hola\_periódico\_lento no se repitió. De acuerdo con lo esperado.

\*\*\*\*\*

### Comando ejecutado:

ps

ps gl | grep [D]emonio

### Explicación:

Con el comando ps se busca saber qué procesos están sin terminar, con ps gl se buscan procesos a mayores que hayan podido quedar colgados.

### Captura:

```
*****
8) Comprobamos que no hay procesos sin terminar.
Esperamos que sólo salgan bash, Ejercicio1.sh y ps
*****

  PID TTY          TIME CMD
  288 pts/0    00:00:00 bash
  422 pts/0    00:00:00 Ejercicio1.sh
  726 pts/0    00:00:00 ps
```

### Análisis:

La salida muestra como sólo quedan presentes bash, Ejercicio1 y ps, lo esperado.

\*\*\*\*\*

### Comando ejecutado:

ls

### Explicación:

Se lanza el comando ls para observar qué archivos quedan después de llevar a cabo el Apocalipsis. Sólo debería quedar Fausto, Demonio y Biblia.

### Captura:

```

*****
9) Comprobamos que no hay ficerhos basura.
Solo deben quedar Fausto.sh, Demonio.sh y la Biblia.txt
*****

Biblia.txt  Demonio.sh  Fausto.sh

```

Análisis:

Efectivamente, la salida devuelve únicamente Biblia, Demonio y Fausto, demostrando que no queda ningún archivo residual.

\*\*\*\*\*

Comando ejecutado:

lslocks | grep flock

cat Biblia.txt

Explicación:

Se lanza el comando lslocks para comprobar si quedan bloqueos pendientes, a mayores y para terminar, se pide que se muestre por pantalla el registro Biblia con cada uno de los procesos marcados en la misma.

Captura:

```

*****
10) Comprobamos que no hay bloqueos pendientes
no debería de salir nada:
*****

```

```

*****
11) Finalmente mostramos la Biblia
*****

18:53:47 [Fausto] -----Génesis-----
18:53:47 [Fausto] El demonio ha sido creado
18:53:47 [Fausto] El proceso 1402 'sleep 10; echo hola >> test1.txt' ha nacido.
18:53:47 [Fausto] El proceso 1415 'yes > /dev/null' ha nacido.
18:53:48 [Fausto] El proceso '1422' 'echo hola_periodico >> test2.txt' ha nacido para ejecutarse per
iódicamente cada 5 segundos.
18:53:49 [Fausto] El proceso '1437' 'echo hola_periodico_lento >> test3.txt; sleep 20' ha nacido par
a ejecutarse periódicamente cada 5 segundos.
18:53:52 [Demonio] El proceso 1415 ha terminado.
18:53:52 [Demonio] El proceso 1415 resucita con pid 1492.
18:53:55 [Demonio] El proceso 1492 ha sido eliminado manualmente.
18:53:55 [Demonio] El proceso periódico 1422 se reencarna con pid 1585.
18:53:58 [Demonio] El proceso 1402 ha terminado.
18:54:01 [Demonio] -----Apocalipsis-----
18:54:01 [Demonio] Iniciando la eliminación de procesos y listas.
18:54:01 [Demonio] El proceso 1585 ha terminado.
18:54:01 [Demonio] El proceso 1437 ha terminado.
18:54:01 [Demonio] Se acabó el mundo.
jorge@GALACTUS:/mnt/c/Users/jorge/Documents/GitHub/University/DyASO/PEC_1$

```

Análisis:

La salida del apartado 10 muestra cómo no quedan archivos con bloqueo pendiente, lo esperado, y la salida 11 nos muestra por pantalla los registros de Biblia, quedando marcados el inicio del demonio, de cada uno de los procesos, cuando terminan, cuando resucitan y cuando se reencarnan, así como el inicio del Apocalipsis su final y la eliminación manual de archivos.

\*\*\*\*\*

## Otros apartados

### Problemas Encontrados y Soluciones

A lo largo de esta PEC me han surgido varios problemas que me llevaron unas cuantas horas de trabajo:

Problema: Manejo de los bloqueos (flock)

Inicialmente, hubo conflictos al acceder simultáneamente a las listas de procesos desde Fausto y Demonio, lo que causaba errores y bloqueos en la consola.

Solución: Busqué múltiples formas de implementar los bloqueos con flock y ajusté los tiempos de espera (-w) para garantizar que no hubiera colisiones al acceder a los archivos. Este enfoque permitió que ambos scripts interactuaran con los recursos compartidos sin conflictos.

Problema: Reencarnación de procesos periódicos

Uno de los mayores desafíos fue implementar la reencarnación de los procesos periódicos, ya que estos debían actualizarse correctamente en las listas y relanzarse al alcanzar su período. En algunos casos, los procesos se reencarnaban varias veces o no lo hacían en absoluto.

Solución: Añadí un control más riguroso en la actualización de las listas, eliminando las líneas anteriores del archivo y reemplazándolas con las nuevas. Además, ajusté los intervalos de tiempo para evitar que los procesos se ejecutaran demasiado rápido.

Problema: Gestión del Apocalipsis

Durante las primeras pruebas, el Apocalipsis no lograba eliminar todos los procesos y recursos, dejando archivos y procesos huérfanos.

Solución: Implementé un barrido final en el que todos los procesos se terminan mediante kill, seguido de la eliminación de los archivos temporales, listas y directorios creados.

## Material Consultado

Para realizar este trabajo, consulté varios recursos que me ayudaron a entender y resolver problemas específicos:

Documentación de Bash:

Me apoyé en la documentación oficial de Bash (man bash) para entender el uso de funciones, redirecciones y estructuras de control.

Manual de flock:

Utilicé el comando man flock y tutoriales en línea para comprender el manejo de bloqueos en scripts de Bash.

Documentación de Linux:

Recurrí a páginas como Studocu donde se pueden encontrar gran cantidad de información y pdf sobre Linux y sus comandos.

Foros y comunidades de programación:

Sitios como Stack Overflow y foros de Linux resultaron útiles para resolver dudas específicas, como el uso avanzado de sed, awk y ps.

Ejemplo del enunciado:

La traza de ejemplo proporcionada en el guión del trabajo fue clave para ajustar los mensajes de salida y garantizar que el programa cumpliera con los requisitos.

## Reflexión Final

Este trabajo fue un desafío interesante que requirió combinar conocimientos de programación en Bash, gestión de procesos en Linux, y sincronización entre scripts. Tuve momentos complicados, especialmente al depurar errores y ajustar tiempos, pero al final conseguí que todo funcionara y que el programa cumpla con los objetivos planteados en el guión. Este proyecto me ayudó a mejorar mis habilidades en scripting y a entender mejor la gestión de procesos en sistemas operativos.

# Bibliografía.

## GNU Bash Reference Manual

URL: <https://www.gnu.org/software/bash/manual/>

Manual oficial de Bash, utilizado para comprender el uso de funciones, redirecciones, y estructuras de control.

## Páginas del manual de comandos:

- `man bash`: Sintaxis y funcionalidades avanzadas de Bash.
- `man flock`: Manejo de bloqueos en archivos.
- `man sed` y `man awk`: Manipulación de texto en Bash.

## Stack Overflow

Comunidad de programadores donde se consultaron dudas puntuales sobre el uso de herramientas de Bash.

URL: <https://stackoverflow.com/>

## Documentación del sistema operativo Linux

Utilizada para comprender la gestión de procesos (`ps`, `kill`, `nohup`) y los comandos relacionados.

URL: [Diseño y Administración de Sistemas Operativos - 71013012 - UNED - Studocu](#)

## Guión de la práctica

Documento proporcionado como parte del material de la práctica, donde se detallan las especificaciones, ejemplos y resultados esperados.