Una de las características históricas del sistema operativo UNIX ha sido un tipo especial de proceso conocido como “demonio” (del inglés daemon ) del sistema, los cuales son utilizados para llevar a cabo ciertas tareas administrativas y para implementar servidores. Este tipo de procesos se distinguen de los procesos comunes principalmente porque no están asociados a una terminal, lo que implica que pueden mantenerse en ejecución incluso cuando el usuario que los inició ha cerrado sesión en el sistema (en el caso de los procesos comunes, estos son terminados automáticamente por el sistema operativo cuando el usuario se desconecta de su terminal de inicio de sesión). Así mismo, estos procesos no poseen flujos de entrada/salida/error estándar, lo que los obliga a comunicarse con los usuarios (usualmente el administrador del sistema) mediante otros mecanismos. Este mecanismo es soportado por todos los sistemas operativos modernos derivados de UNIX, tales como GNU/Linux, Solaris, Darwin, FreeBSD, OpenBSD, entre otros.

En base a esto, en este proyecto se requiere que Ud. desarrolle una aplicación para el sistema operativo GNU/Linux utilizando el lenguaje de programación C, la cual debe cumplir con los siguientes requisitos:

1. La aplicación debe ejecutarse como un proceso tipo demonio (daemon) utilizando llamadas al sistema fork() para tal fin.
2. La entrada de la aplicación se debe realizar unicamente mediante un archivo de texto que debe estar ubicado en la ruta “/etc/proyecto so 1/proy1.ini”. Su proyecto debe identificar que este archivo exista antes de realizar cualquier otra tarea. En caso de que el archivo no exista se debe notificar al usuario (ver siguiente punto) y terminar inmediatamente. El formato del archivo de configuración está definido en la sección de Consideraciones.
3. La salida de la aplicación se debe realizar mediante el servicio de log del sistema operativo utilizando las funciones de biblioteca del paquete syslog.h. Los mensajes de su aplicación deben estar identificados con la etiqueta definida en el archivo de configuración mencionado.
4. La aplicación debe examinar una vez cada cierto tiempo el contenido del directorio “/var/log”, para identificar los cambios que hayan ocurrido en los archivos de log del sistema desde la última revisión. Se deben omitir los subdirectorios y los archivos con extensión “.gz”. Para verificar los archivos debe utilizar el comando de shell “md5sum” mediante las llamadas al sistema fork y exec. Por cada archivo a ser procesado su aplicación debe crear un subproceso el cual se encargar de ejecutar el comando “md5sum”. El resultado del comando debe ser retornado al proceso padre mediante una tubería anónima creada con la llamada al sistema pipe. Se debe crear una tubería por cada subproceso.
5. Con las sumas de verificación de cada archivo el proceso inicial debe determinar cuales de los archivos en cuestión han cambiado con respecto a la revisión anterior,

comparando si la suma del archivo en la revisión anterior es diferente de la suma en la revisión actual (debe asumir que al momento de ejecutar el programa, antes de realizar la primera verificación, la suma de cada archivo es cero. Lo mismo sucede para los archivos que fueron creados entre cada par de revisiones).

6. Con esta lista de archivos modificados, la aplicación debe proceder a empaquetarlos en un único archivo con el siguiente formato: por cada archivo a empaquetar se crea una cabecera que debe contener el nombre del archivo (truncado a 32 caracteres) y el tamaño en bytes del mismo como un entero de 64 bits. Esta cabecera se escribe a un archivo de salida que debe estar ubicado en la ruta “/var/log/PROYECTO SO 1/logs <FECHA>-<HORA>.pak” (FECHA y HORA deben ser sustituidos por valores reales), seguida de todo el contenido byte a byte del archivo en cuestión. Este procedimiento debe realizarse por cada uno de los archivos modificados. Al terminar de empaquetar el último archivo se debe copiar una cabecera especial cuyo nombre de archivo es “FIN” y cuyo tamaño en bytes es 0. Nótese que como el archivo empaquetado se ubica dentro de un subdirectorio de “/var/log” este no debe ser procesado por su aplicación. (Investigue sobre archivos pak para entender mejor este punto).
7. Una vez generado este paquete debe proceder a comprimirlo utilizando el comando “gz” ejecutado mediante las llamadas al sistema fork y exec.

**Característica extra obligatoria:**

Se debe crear un archivo de texto plano extra de salida ubicado en la carpeta raíz del usuario con el nombre cedula1_cedula2.txt o cedula1_cedula2_cedula3.txt. El archivo debe registrar el PID del proceso padre y por cada intervalo los procesos hijos deben registrar su PID en el mismo documento, ejemplo:

PID Padre
Intervalo 0
PID Hijo
.
.
.
PID Hijo

Intervalo n (Donde n es el tiempo en el que fue ejecutado esa rutina, hay que llevar un contador en el tiempo para cada vez que se ejecuta la rutina)

PID Hijo
.
.
PID Hijo

NOTA: Este es un ejemplo, el formato debe estar mas completo según sus consideraciones.

**Consideraciones del proyecto:**

- Utilice el comando “hexdump -C” para verificar que su archivo empaquetado es correcto.
- Puede utilizar la llamada al sistema stat y el macro IS_DIR para identificar y descartar los subdirectorios de “var/log”.
- Si el directorio “/var/log/PROYECTO SO 1” no existe este debe ser creado por su aplicación. Para esto revise la llamada al sistema mkdir.
- El archivo “/etc/proyecto so 1/proy1.ini” es un archivo de texto en formato .INI y debe contener lo siguiente: en la primera linea del archivo se encuentra unicamente el texto “[CONF]”. Las siguientes dos lineas contienen las claves “interval” y ”log tag” las cuales especifican el tiempo que debe esperar su aplicación para hacer una revisión del directorio “/var/log” y la etiqueta a utilizar para la función syslog respectivamente. Un ejemplo del archivo de configuración viene dado en el archivo “proy1.ini.example” incluido con este enunciado.
- Su aplicación debe estar ejecutándose siempre con la permisología del usuario root.

**Consideraciones de los entregables:**

- El proyecto deberá realizarse en grupo de 2 (dos) o 3 (tres) personas máximo.
- La solución debe incluir un Makefile mediante el cual debe ser posible compilar, instalar y desinstalar la aplicación. Un ejemplo de Makefile es provisto junto a este enunciado. Si el código es enviado sin el Makefile o el Makefile no funciona, el código no será evaluado.
- La solución debe incluir un script de shell llamado init.sh para inicializar y terminar el proceso demonio siguiendo el formato establecido por el proyecto Debian. Un ejemplo de tal script es provisto junto a este enunciado.
- Puede dividir su proyecto en tantos archivos .c y .h (unidades de compilación) como necesite. Es obligatorio el uso de cabeceras .h.
- La implementación del proyecto será realizada utilizando el lenguaje de programación C, bajo un sistema Linux, de preferencia compatible con el sistema Ubuntu 24.04 LTS o derivados, debe ser compilable con cualquier versión actual de GCC. Ejemplo: gcc 13.3.0
- Usted deberá entregar el código fuente correspondiente al proyecto totalmente funcional y sin errores. Aquellos códigos que presenten errores de compilación no serán evaluados./home/guoze/Desktop/SO I-2025/Lab 2/Archivos Laboratorio #2 prueba
- Un informe de mínimo 6 páginas en formato .PDF que contenga una descripción del programa desarrollado. Debe describir como está estructurado el código fuente e identificar los puntos de su código fuente donde se implementaron cada funcionalidad.

Se debe anexar cada descripción/explicación con sus respectivas capturas. Además, se deben anexar casos de prueba, su estructura y su resultado en la ejecución de su solución.

- Anexar los archivos de sus casos de prueba con resultados reales.
- No se aceptarán entregas fuera de la fecha y hora establecida.
- De acuerdo con lo establecido en la Ley de Universidades, las copias serán penalizadas con la nota mínima (0) para todos los involucrados.
- Se prohíbe el uso de IA generativas (LLM) para desarrollar el proyecto, de ser detectado la nota será severamente penalizada con la nota mínima (0).
- El asunto del correo debe ser [SO]_[Proy1]_[Cedula1]_[Cedula2] o [SO]_[Proy1]_[Cedula1]_[Cedula2]_[Cedula3], si no cumple con el formato del asunto, la nota será severamente penalizada. De no cumplirse tendrá una penalización de 5 puntos.
- El código debe estar escrito de una manera clara, ordenada y bien documentada/comentada.
- Los entregables del proyecto deben ser enviados en un archivo .zip al correo laboratoriosisop@gmail.com.
- La fecha de entrega esta pautada para el día 21/06/25 hasta las 11:59 PM.

La distribución de puntos queda de la siguiente manera:

- ✓ Informe - 4 pts
- ✓ Defensa individual - 3 pts
- ✓ Documentación/comentarios - 1 pto
- ✓ Funcionalidad del código - 12 pts