Colegio Universitario de Cartago
Tecnologías de la Información
TI-163 Arquitectura y Sistemas Operativos

Tarea programada No.1

Aspectos generales

1) La tarea es individual
2) Sistema Operativo: Linux
3) Lenguaje de programación: C
4) Valor de la tarea 10% de la nota final del curso
5) La documentación solicitada debe incluir como mínimo, en un único archivo (zip, rar):
    a. Código fuente del programa
    b. Archivo con el rastreo de las llamadas al sistema de interés para esta tarea y
    c. El reporte estadístico (o resumen) de las llamadas al sistema

    (Los puntos b y c son determinados con la herramienta **Strace**.)
6) Entrega de la documentación: 01/Febrero/2023 (vía correo electrónico antes de las 12 medio día). El nombre del archivo que se envía por correo **debe ser**: **TP1_G2_*apellidos*.rar**. El documento debe enviarse utilizando el **correo institucional**, no se aceptan documentos de correos personales.
7) La tarea se revisará durante la lección de ese mismo día (01/Feb/2023) en orden alfabético, por apellido del alumno. Las citas para la revisión se muestran al final del presente documento.
8) Es **indispensable** que la tarea se realice con **llamadas al sistema** cuando se manipulen archivos. De esta forma las llamadas al sistema para la manipulación de archivos son las de interés para la parte del Strace.

Introducción
La mayoría de las operaciones de entrada/salida en un SO del tipo UNIX pueden realizarse con las llamadas: `open (openat)`, `read`, `write`, `lseek` y `close`.

Para el kernel del SO, todos los archivos abiertos se identifican utilizando un descriptor de archivo (*file descriptor*). Cuando abrimos un archivo que ya existe, el kernel retorna un descriptor de archivo al proceso. Cuando se desea leer o escribir al archivo, identificamos el archivo con ese descriptor que se obtuvo con la llamada "abrir" (`open`).

Cada archivo abierto tiene una "posición actual" (*current read/write position o "currente file offset"*). Está representado por **un entero no negativo** que mide el **número de bytes desde el principio del archivo**. La mayoría de las operaciones normalmente inician en la "*posición actual*" y crean un incremento en esa posición igual al número de bytes leído o escrito. Por definición, esta posición se inicializa en cero cuando el archivo se abre, a menos que la opción **O_APPEND** se especifique. La

posición actual (*current_offset*) de un archivo abierto puede cambiarse explícitamente utilizando la llamada al sistema `lseek`.

Descripción de la tarea

La tarea consta de **dos partes independientes**.

- La primera es implementar un código, completamente funcional, como se describe a continuación.

- La segunda parte de la tarea es el **seguimiento** de las "llamadas al sistema" **utilizadas por el código implementado en la primera parte** utilizando la herramienta `Strace`.

I parte.

Se debe implementar un código que se ejecutará en la "*Terminal*" del SO Linux. El código a implementar debe leer de manera **parcial o total** un archivo de texto. En el caso de que la lectura sea parcial se le debe especificar al programa tanto el byte de inicio como la cantidad de bytes que debe leer. En el caso de que la lectura sea total se le indicará al programa una letra "T", tal y como se indicará posteriormente. Como resultado de la lectura, el programa debe escribir, en un archivo independiente, pero cuyo nombre también se debe indicar en la línea de comandos al ejecutar el programa, lo siguiente:

a. La cantidad de vocales leídas. Especificando separadamente cada una de ellas.
b. La cantidad total de líneas que contiene el archivo.

Los dos resultados anteriores también deben mostrarse en pantalla.

En el archivo de resultados,cuyo nombre es variable, puede ser histórico (no se eliminan los conteos anteriores) o bien podría ser no histórico, donde sí se deben eliminar los conteos de vocales y total de líneas de "corridas" anteriores. Esto también se indicaría como un parámetro al programa a implementar.

Dado lo anterior, el código de la solución debe ser capaz de aceptar del usuario **cuatro o cinco parámetros (o argumentos)** *en la misma línea de comandos de la terminal* que corresponderán a:

(1) el nombre del archivo que se debe leer.

(2) un número entero positivo que representará la posición de inicio de lectura en el archivo. En el caso de que la lectura sea en todo el archivo este parámetro será una letra "T".

(3) un número entero positivo que representará la cantidad de bytes que deben leerse. En el caso de que la lectura sea en todo el archivo este parámetro se omitirá.

(4) El nombre del archivo de salida donde se indican los resultados.

(5) Si el resultado es o no histórico, indicado con las letras **h** (histórico) ó **Nh** (no histórico).

El archivo a leer será entregado por el profesor el día de la revisión de la tarea programada.

Ejemplo No. 1:

./vocales.out Paradoja.txt 13 82 Res.txt h

Entonces se debe interpretar esa entrada en la “Terminal” de la siguiente manera:

- El nombre del programa ejecutable es **vocales.out**
- El archivo a leer se llama **Paradoja.txt.**
- La **lectura** se debe **iniciar** en el **byte número 13.**
- La **cantidad** de bytes que se deben leer: **82**.
- El archivo de salida tiene por nombre “Res.txt” y debe ser histórico (no se borran los cálculos de corridas anteriores, por lo que los cálculos de cantidad de vocales leídas y total de líneas se agrega al final de esos cálculos previos).

Si el contenido del archivo “Paradoja.txt” fuera el siguiente:

PARADOJAS DE NUESTRO TIEMPO
Hoy tenemos casas más grandes y familias más pequeñas
Más facilidades, pero menos tiempo
Tenemos mayor preparación, pero menos sentido común
Más conocimiento, pero menos discernimiento
Tenemos más expertos, pero más problemas
Más medicinas, pero menos bienestar
Gastamos demasiado. Reímos demasiado poco
Manejamos demasiado rápido. Nos enojamos demasiado pronto
Nos acostamos demasiado tarde. Leemos demasiado poco
Vemos demasiada televisión y rezamos muy rara vez
Aumentamos nuestras posesiones pero disminuimos nuestros valores
Hablamos demasiado, amamos muy poco y mentimos muy a menudo
Hemos aprendido a ganarnos la vida, pero no a vivir.

Como respuesta, **en pantalla y en el archivo Res.txt,** se deberá mostrar lo siguiente:

Cantidad de vocales:
a = 11
e = 8
i = 5
o = 4
u = 2

Total de líneas = 14

Ejemplo No. 2:

./vocales.out Paradoja.txt T Res.txt h

En este caso también se debe leer el archivo “Paradoja.txt” pero se deben obtener los resultados del archivo completo, esto lo indica el parámetro “T”. Los resultados se deben almacenar en el mismo archivo anterior (“Res.txt”), y puesto que se está solicitando que sea histórico (h), además de los datos anteriores, se debe agregar al final del archivo, los datos solicitados para esta lectura del archivo.

//

Es importante que el código a desarrollar tenga la verificación de los parámetros que se ingresan en la línea de comandos, es decir:

- que los archivos a utilizar sean del tipo .txt,
- que los números ingresados como principio de lectura y cantidad de bytes a leer sean enteros positivos mayores a cero.
- que el parámetro de lectura total del archivo, si debe ser histórico o Nohistórico sean efectivamente las letras que se indican en este requerimiento: T, h, Nh

Se debe recalcar que tanto el el archivo de lectura como el de salida pueden tener cualquier nombre, por lo que no deben ser “fijos” en el programa fuente.

II parte.

Utilización del `Strace`.

Como segunda parte de la tarea, **el programa diseñado por el alumno** se debe correr utilizando una utilidad que haga el seguimiento (rastreo) de las llamadas al sistema.

Para lograr el seguimiento de las llamadas al sistema, Linux provee una utilidad llamada `Strace`. Al igual que muchas utilidades de Linux, esta tiene varias **opciones** (parámetros que se le indican a la herramienta) y que permiten entre otras cosas, lo que se solicita para la presente tarea:

1) Mostrar llamadas al sistema específicas (por lo que se **solicita para la presente tarea que se muestren exclusivamente las llamadas específicas utilizadas en la tarea: open (openat), read, write, close y lseek.**

2) Generar un “**reporte estadístico**” (o resumen), que también debe ser parte de la documentación. El reporte estadístico se obtiene del ***Strace como parte de las opciones con que cuenta esta herramienta***, debe ser invocado explícitamente como un parámetro del Strace.

3) Almacenar la ejecución de strace en un archivo (punto b de la documentación solicitada)

`Strace` es una herramienta de **línea de comandos,** útil para diagnosticar, dar instrucciones y ejecutar tareas de depuración.

Trazar llamadas al sistema con strace
La herramienta permite rastrear las llamadas realizadas por el programa que se desea depurar.
La forma de invocar la herramienta sería:

```
strace comando/programa
```

Por ejemplo, la siguiente opción nos permite rastrear todas las llamadas al sistema realizadas por el **comando df**, (df= disk free) para ello ejecutamos lo siguiente:

```
strace df
```

Otro ejemplo podría ser con el comando ls (ls=list files): `strace ls`

Trazar llamadas específicas
La opción **-e** *expr* es una "**expresión calificada"** que modifica cuál elemento debe rastrearse o cómo se deben rastrear.
Para hacer el seguimiento (rastreo) de llamadas al sistema específicas se pueden utilizar comandos como el siguiente:

```
strace -e trace=write df
```

Que muestra, **únicamente**, las llamadas al sistema **write** del comando **df** en salida estándar:

Se podría hacer el rastreo de varias llamadas al sistema a la vez:

```
strace -e trace=write,open df
```

Que muestra, **únicamente**, las llamadas al sistema **write y open** del comando **df** en salida estándar (monitor).

Redireccionar el rastreo a un archivo de salida
Si deseamos escribir los mensajes de seguimiento enviandolos a un archivo, usaremos el -o. Esto significa que solo la salida del comando se imprime en la pantalla de la siguiente manera:

```
strace -o Llamadas_ls.txt ls
```

En este caso le estaríamos indicando al comando "strace" que la salida se redireccione a un archivo, llamado en este ejemplo "Llamadas_ls.txt"

/***/

Para el caso del ejemplo de la tarea, donde el ejecutable se llama “vocales.out” utilizaremos:

```
strace -o salida.txt ./vocales.out Paradoja.txt 13 82 Res.txt h
```

De esta forma la salida se redirecciona (-o) al archivo “salida.txt”, el resto de la línea (a partir de ./vocales.out …..) es el programa al que debe hacerse el seguimiento de llamadas.

Calificación de la tarea:

- Puntualidad : 5 puntos
- Documentación : 5 puntos
- Funcionalidad : 90 puntos (las dos partes de la tarea tienen el mismo valor)

En el caso de la primera parte (código y ejecución del programa) se califica:

1. Que se haga lo solicitado y con los parámetros indicados (60 pts.)
2. Utilización de las llamadas al sistema: open (openat), read, write, lseek, close (10 pts.)
3. Verificación de los parámetros de entrada (10 pts.)
4. Claridad y lógica de la programación. (10 pts.). Entre otros:
    - Métodos lógicos y bien identificados para realizar la taea completa.
    - Claridad al definir variables
    - Comentarios e identación del código

Citas para la revisión de la tarea

Grupo 1
TI-163. Arquitectura y Sistemas Operativos

Fecha: 01/Febrero/2023

Carné	Nombre	Hora
118320964	AGUERO CASTRO JOSTIN ALEXIS	06:00 PM
305440887	BLANCO CALDERON DARREN MANUEL	06:10 PM
119000542	CALDERON LEIVA GEANCARLO	06:20 PM
114050314	CALDERON VELASQUEZ MONICA PAOLA	06:30 PM
117010442	CASCANTE ZELEDON JONATHAN JESUS	06:40 PM
304820019	CERDAS SOLANO DAVID	06:50 PM
117170097	GRANADOS ALVARADO SEBASTIAN	07:00 PM
305270457	HERNANDEZ MENA MARIAM	07:10 PM
305380744	HIDALGO CHACON DIDIER ANDRES	07:20 PM
118400418	LEITON DURAN CRISTOPHER ALEJANDRO	07:30 PM
118480208	MENDEZ POVEDA JOSE PABLO	07:40 PM
305450855	MONGE RAABE ANDREY ALONSO	07:50 PM
305450056	NUÑEZ BARBOZA DAYAN YULEISY	08:00 PM
116060194	OBANDO JIMENEZ MARIA TERESA	08:10 PM
305190829	PÉREZ GONZÁLEZ BRYAN	08:20 PM
305120516	RAMIREZ COTO NATHALIA	08:30 PM
304940352	SALAZAR CUBERO OSVALDO ANTONIO	08:40 PM
503830992	SALGADO MARTINEZ VICTOR JULIO	08:50 PM