Instituto Tecnológico de Costa Rica

Principios de Sistemas Operativos

Profesor Kevin Moraga

Tarea Corta 1

Sebastián Solórzano Guzmán
2016138505

1. Introducción

Esta tarea tiene como objetivo profundizar en los conceptos y funcionamiento de las llamadas al sistema en los sistemas operativos mediante el desarrollo de una aplicación que se encargue de rastrear"las llamadas al sistema de otra aplicación o proceso.
La aplicación recibe por parámetros el proceso a rastrear y esta debe correr el programa y mostrar información acerca de cada llamada al sistema realizada por este. También debe aceptar dos posibles opciones, -v muestra todas las llamadas al sistema hechas por el programa y -V debe hacer lo mismo con la excepción de que esta vez el rastreador debe parar cada vez que se detecte una llamada al sistema y resumir la ejecución del proceso al detectar que se presiona una tecla.

2. Ambiente de desarrollo

Para la implementación de esta tarea se utiliza el lenguaje de programación Rust en conjunto con el editor de texto Sublime 3. Rust es un lenguaje de programación compilado, de propósito general y multiparadigma que está siendo desarrollado por Mozilla. Ha sido diseñado para ser ün lenguaje seguro, concurrente y práctico". Es un lenguaje de programación multiparadigma que soporta programación funcional pura, por procedimientos, imperativa y orientada a objetos.
Según la política de Mozilla, Rust es desarrollado de forma totalmente abierta y busca la opinión y contribución de la comunidad. El diseño del lenguaje se ha ido perfeccionando a través de las experiencias en el desarrollo del motor de navegador Servo, y el propio compilador de Rust. Aunque es desarrollado y patrocinado por Mozilla y Samsung, es un proyecto comunitario. Una gran parte de las contribuciones proceden de los miembros de la comunidad.

3. Estructuras de datos usadas y funciones

Se hace uso de 3 funciones para la implementación de esta tarea:
main(): Esta es la función principal del programa, es la función por defecto por la que empieza Rust a la hora de ejecutar un programa y en este caso se utiliza para obtener los parámetros pasados por línea de comando, así como para validar si se especifica una opción y proceder a la función correspondiente a esta.
trace(mut args: Vec<String>): Esta función se encarga de implementar la funcionalidad de la opción -v especificada en las indicaciones de la tarea. Recibe un vector con el resto de los argumentos obtenidos en el main, de estos obtiene el programa a ejecutar y sus argumentos para iniciar la ejecución del mismo haciendo uso de la biblioteca ptrace para obtener la información de las llamadas al sistema utilizadas por el programa.
halted_trace(mut args: Vec<String>): Esta función corresponde a la funcionalidad de la opción -V. Recibe el mismo vector que trace y lo utiliza para iniciar la ejecución del programa, con la diferencia de que esta función detiene la ejecución del programa con la detección de una llamada al sistema y continua la misma con la detección de una tecla.

4. Instrucciones para ejecutar el programa

Para ejecutar el programa se hace uso de la herramienta Cargo de Rust, esta es una herramienta de manejo de paquetes y anfitrión de crates para el lenguaje de programación Rust. El programa se ejecuta con el comando cargo run – [opciones rastreador] prog [opciones de prog] ejemplos:

cargo run – -v ls -al
cargo run – -v ls
cargo run – -V ls -al
cargo run – ls -al

5. Actividades realizadas por estudiante

Instalación y familiarización con el lenguaje de programación Rust 3 horas 9 Marzo.

Investigación sobre ptrace 4 horas 10 y 11 Marzo.

Obtención de parámetros por línea de comando 1 hora 30 minutos 12 Marzo.

Ejecutar programa y utilizar las funciones de ptrace 4 horas 14 Marzo.

Obtención de información sobre llamadas al sistema 3 horas 16 Marzo.

Documentación 1 hora 30 minutos 16 Marzo.

Total de horas: 17.

6. Autoevaluación

Al final el programa logra recibir los parámetros y ejecutar el proceso pero no es capaz de obtener toda la información sobre las llamadas al sistema realizadas por este. Es capaz de obtener cierta información de los registros acerca de las llamadas al sistema pero esta no es clara y no se logra obtener todas las llamadas.
A la hora de buscar información para poder completar esta tarea me topé con 2 situaciones, Rust cuenta con una muy buena documentación e incluso un tutorial muy útil para poder comenzar a programar en este lenguaje, sin embargo a la hora de investigar sobre la biblioteca ptrace se contaba con muy poca documentación y falta de ejemplos acerca de su uso, por lo que hace falta referirse al manual de linux para ver el funcionamiento de la llamada al sistema ptrace.
A pesar de ciertas limitaciones de tiempo debido a exámenes y tareas de otros cursos igual siento que pude haberle dedicado más tiempo a esta tarea.

Evaluación:
opción -v: 0
opción -V: 0
ejecución de prog: 20
análisis de syscalls: 5
documentación: 15

7. Lecciones aprendidas de la asignación

De esta tarea me quedo con el uso del lenguaje de programación Rust, es cuál me parece un buen lenguaje, cómodo de aprender, con buena documentación aunque no se puede comparar a lenguajes mucho más viejos y con comunidades mucho más grandes como C o Java.
También me queda el conocimiento acerca de las señales y llamadas al sistema, en concreto el uso de ptrace para rastrear llamadas al sistema realizadas por un proceso, aunque su funcionamiento en Rust no me queda del todo claro.

8. Bibliografía

*Rust (lenguaje de programación). (s.f.). Accesado el 16 de Marzo del 2021. Recuperado de: https://es.wikipedia.org/wiki/Rust_(lenguaje_de_programaci\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\endgroup\relax\let\ignorespaces\relax\accent19o\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\protect\penalty\@M\hskip\z@skipn).