Ejercicio 5: Estado

Materia: Computación tolerante a fallas

Universidad de Guadalajara

Profesor: Michel Emanuel López Franco

27/02/2024

José Maximiliano Díaz Méndez

Introducción

Realizar un programa que sea capaz de revisar el estado de tu aplicación.

Requisitos

Para poder compilar y ejecutar los programas que utilice para la tarea es necesario Rust (Lenguaje) y cargo (Administrador de paquetes).

Como escanear

Todos los procesos

Usando el comando `cargo run --bin proc-manager -- scan` es como podemos obtener el listado de procesos.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-5 ○  b391e0a|master ⚡
>  cargo run -q --bin proc-manager -- scan
PID: 14557        Name: trustd    Status: Runnable
PID: 1636       Name: chrome_crashpad_handler   Status: Runnable
PID: 95324      Name: com.apple.WebKit.WebContent       Status: Runnable
PID: 23443      Name: PlugInLibraryService      Status: Unknown
PID: 60432      Name: containermanagerd         Status: Runnable
PID: 23304      Name: transparencyd     Status: Runnable
PID: 527        Name: powerd    Status: Unknown
PID: 878        Name: com.apple.DriverKit-IOUserDockChannelSerial       Status: Unknown
PID: 1   Name: launchd   Status: Unknown
PID: 882        Name: distnoted         Status: Unknown
```

Procesos por nombre

Usando el comando `cargo run --bin proc-manager -- scan dummy` es como podemos obtener el listado de procesos que contienen `dummy` en su nombre.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-5 ○  b391e0a|master ⚡
>  cargo run -q --bin proc-manager -- scan dummy
PID: 20694    Name: dummy-process     Status: Runnable
```

Procesos por PID

Usando el comando `cargo run --bin proc-manager -- scan 123` podemos obtener el proceso con el PID especificado si es que existe.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-5 ○  b391e0a|master ⚡
>  cargo run -q --bin proc-manager -- scan 20694
PID: 20694    Name: dummy-process     Status: Runnable
```

Como matar procesos

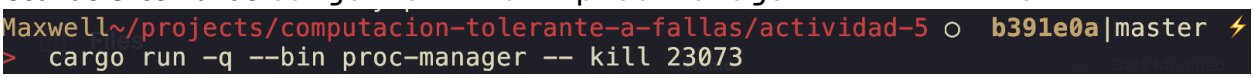
Matar por nombre

Usando el comando `cargo run --bin proc-manager -- kill dummy-process`.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-5 ○  b391e0a|master ⚡
>  cargo run -q --bin proc-manager -- kill dummy
```

Matar por PID

Usando el comando `cargo run --bin proc-manager -- kill 123`.



Conclusión

Al principio busqué alguna programa de CLI para poder ver los procesos para crear el programa usándolo dando con el comando *ps* pero solo funciona en sistemas Unix-like por lo cual desistí y decidí buscar mejor librerías para Rust que fue el lenguaje que decidí utilizar, aquí me puse a revisar la lista de dependencias de *bottom* un administrador de procesos escrito en Rust para terminal que utilizo y vi que usaba la librería *sysinfo* la cuál es multiplataforma (Hasta cierto punto) y fue la que decidí utilizar al final para el programa. Realmente no tuve grandes dificultades para lograrlo más allá de que apenas estoy aprendiendo a utilizar Rust y este tiene muchas particularidades y diferencias a C y C++.

Referencias

Gomez, G. (s. f.). Crates.io: Rust package Registry. crates.io: Rust Package Registry. Recuperado 26 de febrero de 2024, de https://crates.io/crates/sysinfo