Universidad San Carlos de Guatemala

Facultad de Ingeniería

Escuela de ciencias y sistemas

Laboratorio Sistemas Operativos 2

Primer semestre 2023

Aux. Sergio Estuardo Cruz S.

Manual - Técnico
Proyecto 3 _ 4

Julio José Orellana Ruíz

201908120

Índice

Introducción

Este documento tiene como objetivo proporcionar una guía detallada para la implementación de la fase dos de un proyecto en el que se busca monitorear el sistema de una computadora y administrar sus puertos USB. En esta fase, se requiere que se agregue el monitoreo de la memoria RAM y se cree un programa para administrar los puertos USB, que solicite autenticación para su uso. Este programa bloqueará el acceso a los puertos USB, permitiendo su liberación solo a través del uso de un usuario y una contraseña específicos. Además, el programa llevará una bitácora de los archivos copiados desde y hacia los puertos USB. Se espera que, al finalizar la lectura de este documento, los usuarios tengan una comprensión completa de cómo implementar estas características en el sistema de una computadora.

Este documento está dirigido a desarrolladores y otros profesionales técnicos que trabajan con sistemas informáticos y que desean implementar estas soluciones en su entorno de trabajo.

Administrador de tareas de Linux

Requerimientos Sistema Operativo

Este software fue desarrollado con una maquina Ubuntu que cubre las siguientes especificaciones:

- Ubuntu 22.04.2 LTS
- Sistema de 64 bits

El software no está delimitado para que versiones puede funcionar de Ubuntu Linux, pero se espera que funcione con al menos la ultimas 4 versiones de Ubuntu LTS:

- Ubuntu 22.04.2
- Ubuntu 20.04 LTS
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS

Pagina de ubuntu: https://ubuntu.com/

Requerimientos mínimos del sistema

- Procesador de 1 GHz (por ejemplo, Intel Celeron) o mejor.
- 1.5 GB RAM (Memoria del sistema).
- 7 GB de espacio libre en la unidad interna para la instalación.
- Una unidad de DVD o un conector USB para el medio de instalación.
- Acceso a internet (para instalar actualizaciones durante el el proceso de instalación)

Programa en go para memorias

Este es un programa en lenguaje Go que permite al usuario seleccionar si desea otorgar o no permisos de escritura a una unidad USB conectada a la computadora. Además, el programa utiliza la herramienta "inotifywait" para detectar y mostrar en tiempo real las acciones que se realizan en la unidad USB.

```go
package main

import (
	"fmt"
	"os/exec"
)
```

La declaración del paquete "main" indica que este es el programa principal que se ejecutará y se importan los paquetes necesarios para el programa: "fmt" para imprimir mensajes en pantalla, y "os/exec" para ejecutar comandos en la línea de comandos del sistema operativo.

```go
// Ejecuta el comando "sudo su"
cmd1 := exec.Command("sudo", "su")
if err := cmd1.Run(); err != nil {
	fmt.Println("Error al ejecutar sudo su:", err)
	return
}
```

Se ejecuta el comando "sudo su" para obtener permisos de superusuario en el sistema operativo. En caso de que ocurra un error, se muestra un mensaje de error en pantalla y se termina la ejecución del programa.

```go
for opcion != 3 {
	fmt.Println("Escriba la opcion que desea elegir: ")
	fmt.Println("1. Dar permisos")
	fmt.Println("2. No dar permisos")
	fmt.Println("3. Salir")

	fmt.Scanln(&opcion)

	switch opcion {
	case 1:
```

Ese es el funcionamiento del Menu.

Al entrar al caso 1:

```
case 1:
        cmd := exec.Command("sudo", "chmod", "777", "/media")
        output, err := cmd.Output()

        if err != nil {
                fmt.Println("Error al ejecutar el comando opcion 1:", err)
                return
        }
```

En el caso 1 se le dan todos los permisos a la carpeta /media que es la que maneja los puertos de entrada. Si hay un error lo va a reportar al contrario ya se puede ingresar la memoria y ingresar a sus datos.

```
// Observando la USB
fmt.Println("Observando la USB...")
fmt.Println("Combinacion de teclas 'ctrl + c' para detener la observación de la USB...")
for {
        cmd2 := exec.Command("inotifywait", "-e", "create,delete,move", "/media/julio-or/UBUNTU 20_0")
        output, err := cmd2.Output()
        if err != nil {
                fmt.Println("Error al ejecutar inotifywait:", err)
                return
        }
        fmt.Println("Se detectó una acción en la USB: ", string(output))
}
```

Luego se queda observando la USB cualquier cambio que suceda en la usb. La ruta que se muestra es el puerto que se utiliza y esta dando su información en forma de bitácora.

Caso 2:

```
case 2:
        cmd := exec.Command("sudo", "chmod", "000", "/media")
        output, err := cmd.Output()

        if err != nil {
                fmt.Println("Error al ejecutar el comando:", err)
                return
        }

        fmt.Println("Has seleccionado la opción de no dar permisos: ", string(output), "\n")
```

Se le quitan todos los permisos a la /media que serian todos los accesos de usb. Y si intentan ingresar a ver los archivos no deja.

Caso 3:

```
case 3:
        fmt.Println("\nGracias por usar nuestro programa. ¡Hasta luego!")
```

Por si se le ingresa a la opción del menú salir, termina el programa.

Default:

```
default:
        fmt.Println("Opción inválida. Por favor, seleccione una opción válida.\n")
}
```

Es por si se ingresa opciones que no estén en el menú

WAILS Framework

Wails es un framework de desarrollo de aplicaciones de escritorio multiplataforma que combina la tecnología web (HTML, CSS, JS) con el lenguaje de programación Go.

Está diseñado para ser fácil de usar y permite a los desarrolladores crear aplicaciones de escritorio modernas y altamente personalizables con una interfaz de usuario basada en la web.

Una de las principales características de Wails es que utiliza una capa de presentación web para crear la interfaz gráfica de usuario (GUI), lo que permite a los desarrolladores usar las habilidades que ya tienen en tecnologías web para crear aplicaciones de escritorio.

Además, Wails proporciona acceso completo a las API nativas de la plataforma, lo que significa que los desarrolladores pueden crear aplicaciones de escritorio que se integren completamente con el sistema operativo y acceder a sus recursos, como archivos, ventanas y otros elementos del sistema.

Instaladores para usar WAILS

Para este proyecto se usó la versión de **wails 2.3.1**

- Go versión 1.18 o superior
- Node js 14 o superior
- Wails 2.3.1 o superior

Al tener instalado Go y declarado en las variables PATH, se procede a ejecutar en la consola de comandos:

- go install github.com/wailsapp/wails/v2/cmd/wails@latest

al instalarlo en la misma consola para verificar la instalación se hace:

- wails doctor

Esto genera una lista de todo lo que ya esta. A veces tira una lista de ultimo que dice que hay que instalar unos paquetes, esto solo pasa si no tenia nada instalado. Serian:

- Instalación de dependencias de librerías gráficas (como GTK o Qt)
- Instalación de herramientas de compilación (como GCC o Clang)
- Kubernetes
- Gnsn

Al estar completo le dira que esta listo para el uso.

Correr el Proyecto de WAILS

Al ejecutar el proyecto se necesita el comando:

- **Wails dev** este comando sirve para levantar el servidor y el front-End y está en modo desarrollo por lo que cualquier cambio se ira viendo inmediatamente.

Modificar WAILS

Para modificar wails necesitamos saber que hay ciertas configuraciones importantes para su ejecución, las cuales son:

Archivo app.go: Este archivo se encuentra en la raíz del proyecto y contiene el código principal de la aplicación en Golang. Aquí es donde se pueden agregar nuevas funciones, modificar las existentes y configurar la aplicación.

App.js: En este archivo se tiene que mandar a llamar los métodos creados desde la app.go. claro esta que se tiene que seguir una estructura la cual se explicara más adelante.

App.jsx: esto ya es parte de react pero aca se manejo lo que es el entorno grafico de la pantalla con la lagoica del app.

Carpeta frontend: Esta carpeta contiene todo el código relacionado con la interfaz de usuario de la aplicación. Aquí es donde se pueden agregar o modificar componentes HTML, CSS y JavaScript.

Archivo wails.json: Este archivo es el archivo de configuración principal de Wails. Aquí se pueden configurar varias opciones, como el título de la ventana, el ícono de la aplicación, el tamaño de la ventana, etc.

Carpeta assets: Esta carpeta contiene todos los recursos que se pueden utilizar en la aplicación, como imágenes, fuentes, iconos, etc.

Carpeta bindings: Esta carpeta contiene los archivos de código que se utilizan para comunicar el frontend de la aplicación con el backend en Golang.

Carpeta cmd: Esta carpeta contiene todos los comandos personalizados que se pueden ejecutar desde la terminal.

Estructuras de archivos WAILS

App.go

Esto son las que no hay que modificar si se va a modificar o se van a trabajar métodos similares.

```go
package main

import (
	"context"
	"fmt"
	"syscall"
)

// App struct
type App struct {
	ctx context.Context
}

// NewApp creates a new App application struct
func NewApp() *App {
	return &App{}
}

// startup is called when the app starts. The context is saved
// so we can call the runtime methods
func (a *App) startup(ctx context.Context) {
	a.ctx = ctx
}
```

/frontend/wails/go/main/APP.js

Este se manda a llamar los métodos creados desde el APP.go. y se le manda si tiene parámetros o no.

```js
// @ts-check
// Cynhyrchwyd y ffeil hon yn awtomatig. PEIDIWCH Â MODIWL
// This file is automatically generated. DO NOT EDIT

export function CPU() {
  return window['go']['main']['App']['CPU']();
}

export function DISK() {
  return window['go']['main']['App']['DISK']();
}

export function Greet(arg1) {
  return window['go']['main']['App']['Greet'](arg1);
}
```

Teniendo en cuenta los archivos que tienen que mantener esta estructura se pueden crear en el backend y extraerlo de manera sencilla y para llamarlo en el front se realiza:

```
import {Greet, CPU, DISK} from "../wailsjs/go/main/App";
import { Pie } from "react-chartjs-2";
```

Si se exporta los módulos que se crean para consumir.

Métodos y configuraciones creadas

DISK

Este método es para obtener da el porcentaje de la RAM utilizada, en este método lo que obtiene es el total de RAM, la libre y lo que se esta utilizando y luego se devuelva la utilizada

```
func getMemoryUsage() (float64, error) {
        var info syscall.Sysinfo_t
        err := syscall.Sysinfo(&info)
        if err != nil {
                return 0, err
        }

        total := info.Totalram * uint64(info.Unit)
        free := info.Freeram * uint64(info.Unit)
        used := total - free
        usage := float64(used) / float64(total) * 100

        return usage, nil
}
```

El método maestro retorna el almacenamiento que esta en uso luego de llamar al método, el cual lo retorna por el tipo Float64.

```
func (a *App) DISK() float64 {
        usage, err := getMemoryUsage()
        if err != nil {
                fmt.Println("Error:", err)
                return 0
        }
        return usage
}
```

App.js

Aca mandamos a llamar los métodos tal como vendrían desde el bakcend

```
export function DISK() {
  return window['go']['main']['App']['DISK']();
}
```

App.jsx

Se usaron esas importaciones para poder manejar Char.js para graficas y el resto para la lógica.

```
import React, {useState, useEffect} from 'react';
import './App.css';
import {Greet, CPU, DISK} from "../wailsjs/go/main/App";
import { Pie } from "react-chartjs-2";
import { Chart as ChartJS, ArcElement, Tooltip, Legend, CategoryScale, registerables } from 'chart.js';
```

Aca le damos la forma recursiva por medio de react de la RAM, para que pueda estar repitiéndose cada cierto momento. El tiempo que se le da que esta expresado en mil, es el tiempo reducido a una milésima que sería un segundo. Eso se tardará para estar actualizando.

```
// ------------ RAM
const [disk, setdisk] = useState('');
const updateResultDisk = (result) => setdisk(result);

useEffect(() => {

    setInterval(() => {
        DISK().then(updateResultDisk)
    }, 5000);
})
```

En estas constantes se llena los datos de la grafica y se les da la configuración de como se mostrara la grafica

```
const dataDisk = {
    labels: ["RAM", "NO RAM"],
    datasets: [
        {
            label: '% de RAM',
            data: [disk, (100-disk)],
            backgroundColor: [
                'rgba(255, 99, 132, 0.2)',
                'rgba(54, 162, 235, 0.2)',
            ],
            borderColor: [
                'rgba(255, 99, 132, 1)',
                'rgba(54, 162, 235, 1)',
            ],
            borderWidth: 1,
        },
    ],
};
```

```
var optionsCpu = {
    responsive: true,
    maintainAspectRatio: false,
    title: {
        display: true,
        text: 'Gráfica de Pie'
    },
    legend: {
        display: true,
        position: 'bottom',
        labels: {
            fontColor: '#333',
            fontSize: 12
        }
    }
};
```

Para el return de la APP se manda a llavar en html para que muestre los datos y aca se les termina de dar la configuración de la RAM.

```
return (
    <div >

        <br />
        <br />
        <br />

        <div id="title">
            Grafica de PIE, Uso de RAM
        </div>

        <div >
            <Pie
                data={dataDisk}
                options={{
                    responsive: true,
                    maintainAspectRatio: false, // Agrega esta opción para deshabilitar la relación de aspecto predeterminada
                    width: 500, // Especifica el ancho de la gráfica
                    height: 500, // Especifica la altura de la gráfica
                    title:{
                        display:true,
                        text:'Porcentaje de la RAM',
                        fontSize:20
                    },
                    legend:{
                        display:true,
                        position:'right'
                    }
                }}
            />
        </div>
        <div>
            RAM =  %{disk}
        </div>
```

Conclusión

En conclusión, este manual técnico describe una solución diseñada para facilitar la transición de los nuevos usuarios de Windows al entorno gráfico de Linux, sin necesidad de conocimientos avanzados de línea de comandos. La solución se basa en el lenguaje de programación Golang y utiliza el framework Wails para mostrar estadísticas de uso del sistema en modo gráfico. Este manual técnico proporciona instrucciones detalladas sobre cómo instalar, utilizar y personalizar la herramienta para satisfacer las necesidades específicas de los usuarios. Se espera que después de leer este manual, los usuarios puedan utilizar eficazmente la solución y adaptarla a sus entornos de trabajo. Este manual está dirigido a administradores de sistemas, desarrolladores y otros profesionales técnicos que trabajan con sistemas Linux.

bibliografía

https://ubuntu.com/

https://wails.io/changelog/

https://go.dev/doc/devel/release

https://nodejs.org/es/download/releases/