TSI

TRANSPORT AND
TELECOMMUNICATION
INSTITUTE

Operating Systems and System Programming

Topic: Working with the File System and System Registry

Student's name: Artjoms
Student's surname: Voroncovs
St. code: st83663

RIGA

TABLE OF CONTENTS

1. Objective

The goal of this task is to develop a console-based system information tool that:

- Displays core system information such as operating system name, version, computer name, and current time.
- Allows the user to browse through the file system to view directories and files with their characteristics.
- Stores user settings, such as the default directory to open when program starts and chosen color scheme.
- Provides a simple and user-friendly terminal-based interface.

The implementation was done using C++ with the ncurses library for UI rendering and POSIX APIs for system and file information.

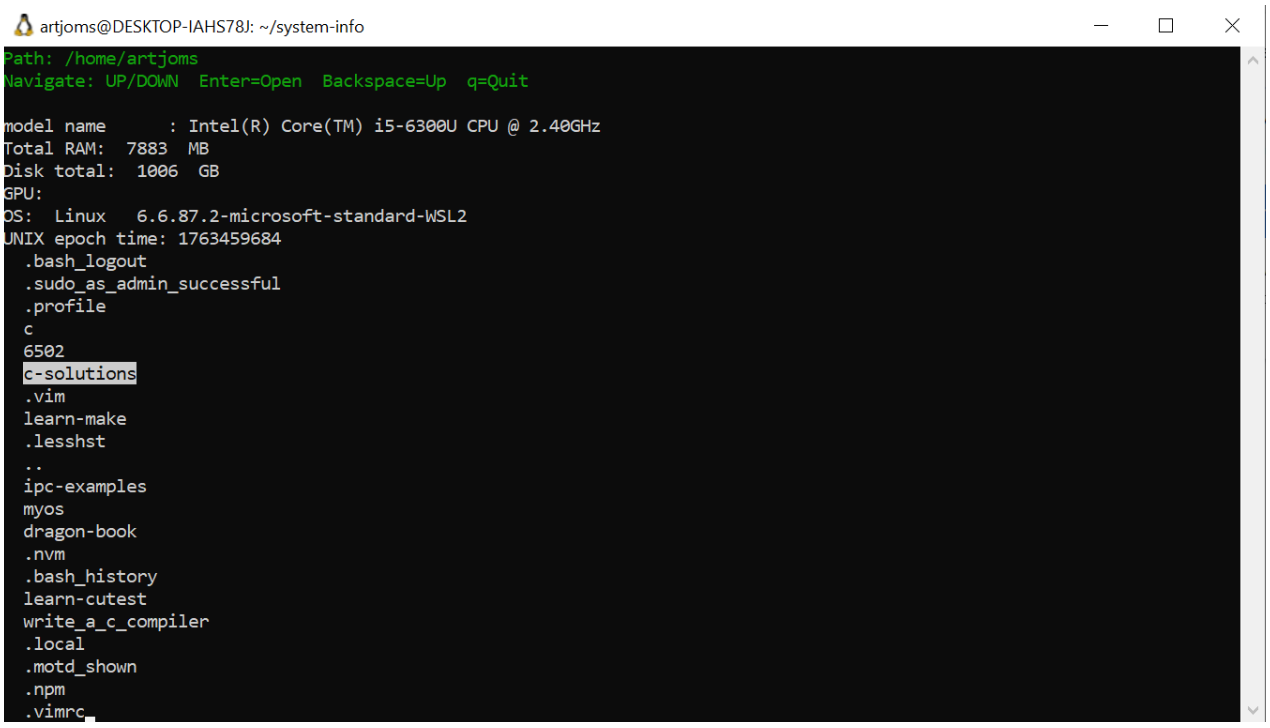
2. Functionality

2.1 System Information Display

System information is gathered by a dedicated class SysInfo that reads system data from Linux virtual files and system calls.
It provides methods such as:

- getCpuInfo() – extracts CPU model name from /proc/cpuinfo.
- getMemInfo() – reads /proc/meminfo to display total and available memory.
- getDiskInfo() – uses statvfs() to show free and total disk space.
- getGpuInfo() – reads GPU details from /sys/class/drm/.
- getKernelInfo() – retrieves kernel name and version using uname().

Example screenshot:



artjoms@DESKTOP-IAHS78J: ~/system-info

```
Path: /home/artjoms
Navigate: UP/DOWN  Enter=Open  Backspace=Up  q=Quit

model name      : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
Total RAM:  7883  MB
Disk total:  1006  GB
GPU:
OS:  Linux   6.6.87.2-microsoft-standard-WSL2
UNIX epoch time: 1763459684
  .bash_logout
  .sudo_as_admin_successful
  .profile
  c
  6502
  c-solutions
  .vim
  learn-make
  .lesshst
  ..
  ipc-examples
  myos
  dragon-book
  .nvm
  .bash_history
  learn-cutest
  write_a_c_compiler
  .local
  .motd_shown
  .npm
  .vimrc
```

2.2 File System Interaction

The application allows users to navigate the file system interactively.
It uses the standard POSIX functions opendir(), readdir(), and stat() to list directory contents.
Each file and folder name is dynamically loaded and displayed in the ncurses interface.

Navigation keys:

- **UP/DOWN arrows** – move selection.
- **Enter** – open a directory.
- **Backspace** – go one level up.
- **q** – quit the program.

The current path is displayed at the top of the screen, and the interface refreshes dynamically as the user navigates.

2.3 Drive and Folder Information

The program lists available logical drives (mounted file systems) and display folders inside logical directories.

2.4 Interface Design

The interface uses the ncurses library to create a text-based graphical environment with the following features:

- Highlighted selection using A_REVERSE.
- Colored text using init_pair() for different color schemes.
- Clear and organized layout that separates navigation instructions, path, system information and file list.

3. Settings Storage

3.1 Implementation

The program saves settings such as the **default path** and **color scheme** in a configuration file located at:

<PROGRAM BIN DIR>/.systeminfo.conf

Example file contents:

```
default_path=/home/artjoms/projects

color_scheme=2
```

On startup, the program reads this configuration file. If it does not exist, default values are used (the current working directory and default color).
Before exiting, it updates the configuration file with the current settings using standard file I/O (fopen(), fprintf(), fclose()).

3.2 Justification

The configuration file approach was chosen because:

- It is simple and human-readable.
- It provides easy persistence across sessions.
- It requires no external dependencies or complex data formats.
- Users can manually edit or reset the file if needed.

4. Code Quality

4.1 Structure

The code is organized into separate modules for clarity and maintainability:

- UI class – handles all ncurses interface logic and user input.
- SysInfo class – provides system and hardware information.
- main.cpp – initializes settings and starts the main user interface loop.

This separation of concerns improves readability and simplifies future modifications.

4.2 Readability and Comments

Each function and section of code contains clear comments describing its purpose and behavior.
Example:

```
// Move selection up in the file list

case KEY_UP:

    if (current_selection > 0) current_selection--;

    break;
```

Variable names are descriptive, and the logic follows a consistent style.
Memory management is handled properly by freeing old directory entries before loading new ones.

5. Report and Justification

5.1 Design Choices

Component	Decision	Justification
Programming Language	C++	Provides object-oriented structure and low-level system access.
UI Framework	ncurses	Portable, lightweight, and ideal for text-based interactive applications.
File System Interaction	POSIX API	Reliable and compatible with all Linux distributions.
Configuration Storage	Plain text file	Simple, transparent, and user-editable.
Information Sources	/proc and uname()	Reliable Linux system information interfaces.
Program Architecture	OOP with UI and SysInfo classes	Encourages modularity and separation of logic.

5.2 Algorithmic Approach

- Directory traversal uses iterative navigation (non-recursive) to maintain control and prevent excessive stack use.
- Each directory change reloads file data dynamically.
- Memory allocation and deallocation are handled carefully to prevent leaks.
- The interface redraws the entire screen for consistency after each user action.

5.3 Reflection

This project demonstrates a practical understanding of:

- Low-level Linux system calls and file system APIs.
- Terminal-based UI design using ncurses.
- Persistent settings management.
- Application structure and modularity using C++.

5.4 Future Improvements

Possible future enhancements include:

- Displaying additional file information (size, permissions, modification time).
- Using JSON or INI format for settings.
- Adding mouse support in the ncurses interface.
- Expanding the program to support Windows using the WinAPI and registry queries.

6. Summary

The “System Info” application fulfills all key requirements of the assignment:

- It accurately retrieves and displays system and hardware information.
- It provides smooth file system navigation within a terminal UI.
- It supports persistent user settings in a configuration file.
- The codebase is clean, modular, and well-commented.
- The design decisions are justified