CMPE 322 – Operating Systems / Project 1 Video Game Console

Deadline: 30.11.2024 23.59

1) Project Definition

In this project a video game console with some games is requested. The game console screen will be your shell screen, games are compiled C programs (each with separate main function, as standalone programs) and playable itself, a main screen to select games, and a logical device which treats like old plug-and-play disks to your console controller. Collecting all of these components will result in a fully functional virtual video game console.

2) Project Parts

2.a) Games

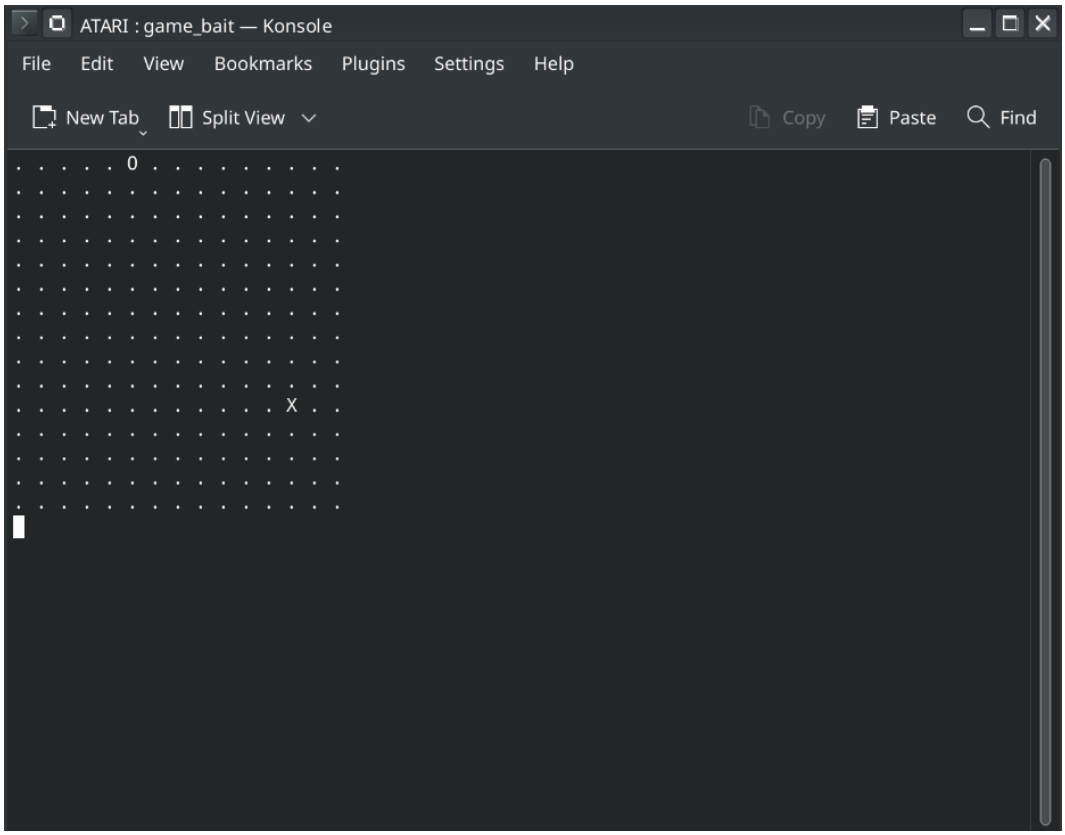
Three different games will be implemented. Each game must be written in C. They have to take input without disturbing the input screen and without pressing enter. Commands must be provided correctly and processes without any error. Also the exit procedure must be smooth. Whenever we decide to exit program gracefully or with interrupt the console must remain intact and there must not be any difference before starting the game program.

There are several rules:

- Games must be interactive. It must take input from user and process it accordingly.
- “q” button is universal exit button. Whenever we press q, the program must exit gracefully. Designing the graceful exit procedure is your task.
- Be careful about memory leaks. In your code use dynamic memory allocation functions (malloc, calloc etc.) at least 1 time, and free it correctly. It will be evaluated.
- General user input keys are “w”, “a”, “s”, “d”. While designing your game use these keys, that is not a must but you should do that. If any other keys are required for your programs, do not hesitate to using them (For example “space” to shoot in tank-shooting game).
- Game names must follow the convention “game_<game-name>” convention. For example, for Tetris game compile your program as “gcc -o game_tetris tetris.c”.

- All games are designed as separate standalone C programs, that is, every game have its own main function. Designing multiple games in a single C file will be penalized.
- All games must exit gracefully with signals “SIGINT” and “SIGTERM”. Any unusual behavior of your shell after exiting with those signals will be penalized.
- Game screen will have a predefined resolution. (For example 15 columns, 20 rows)
- Game screen must show only the video-game console stuff (game, score table, information text etc.). Any other elements seen on the terminal will be penalized. (Except the own features of terminal, such as input cursor bar)

Example Image:



You must implement 3 different games. One of them must be “Snake Game”, the others are up to you. Use your imagination skills and creativity.

Snake game rules are:

- Snake must start in the middle of the grid. (Divide the border lengths by 2, do not try to make complicated calculations).
- Bait must spawn at a random location. The spawn point must be different that snake’s actual position.
- Hitting the border of the map or snake’s own tail is not its end. Just make the snake wait until a new valid input. Do not kill the game.
- Snake’s head will be “O”.
- Snake’s tail will be “#”.

- Bait will be “X”.
- Empty spaces will be denoted by “.”.
- “w” → Go upwards.
- “a” → Go left.
- “s” → Go down.
- “d” → Go right.
- “q” → Exit gracefully.
- Row count: 15
- Column count: 15

Other two games are up to you. Do not try to implement extremely complicated games, it will consume enormous amount of your time.

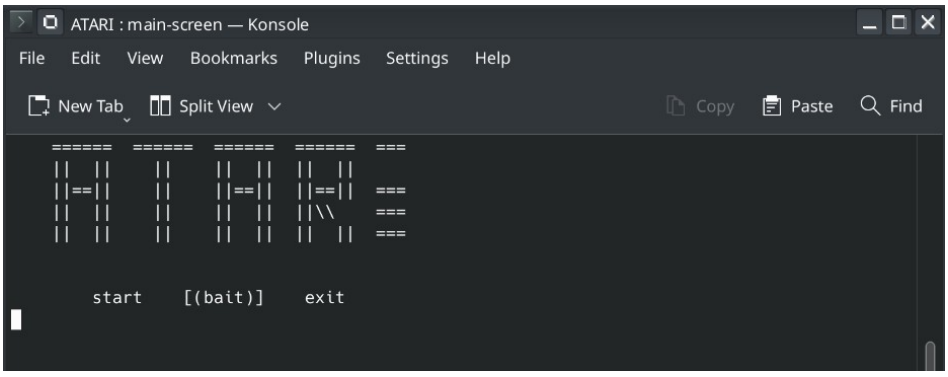
2.b) Console Main Screen

Other that launching those games directly, you must provide a basic main menu screen of your video-game console. This main screen design is up to you.

The main screen must include following features:

- The main screen must provide an understandable interface to select and start games.
- It must scan all executables that their names are starting with “game_” in the main-screen’s directory. For example I design 3 games and their file names are “game_tetris”, “game_snake”, and “game_tank”. The main screen must allow you to choose one of those games and play.
- On the selection screen only the currently chosen game’s name must be displayed. By using “w” and “s” buttons the player must be able to switch between games, and by pressing enter button the selected game must start.
- “q” button will terminate the main screen. Also add an exit button to your main screen and button selection mechanism with “a” and “d” buttons (“a” for left and “d” for right).
- By providing all these features, design of your main screen is up to you. Design an easy-to-use and easily-understandable main screen.
- It must also exit gracefully whenever it receives “SIGINT” and “SIGTERM” signals.
- You do not have to design an extremely well organized main screen. Just implement a proper screen.
- Main menu screen must show only the main-menu (buttons, information text, selection, etc.). Any other elements like “wasd” keys or interrupt ^C string seen on the terminal will be penalized. (Except the own features of terminal, such as input cursor bar)
- Its name must be “main-screen”.

Example Main Menu Screen:



2.c) Disk Device

As you know, we design a virtual video-game console. In this project, we will add plug-and-play feature to our console.

To design a proper virtual disk device:

1. Create a disk image file. It must be formatted as “ext4”.
2. Make it mountable. Use device files. Whenever we want to use / remove it, we must be able to mount / unmount.
3. Place your main menu and game executable files into the your newly-created virtual disk.

By accomplishing these steps, you can actually transfer your games to another user’s computer by only copying the image. The receiver will get all of the games (Like transferring file with flash-disk).

3) Project File Hiearchy

```
.
├── storage_vgc.img
├── bin
│   ├── game_1
│   ├── game_2
│   ├── game_3
│   └── main-screen
├── <device-file> -> /dev/<device-file>
├── mount
├── src
│   ├── src_1.c
│   ├── src_2.c
│   ├── src_3.c
│   └── main-screen.c
├── initialize.sh
├── purge.sh
├── terminate.sh
└── startup.sh
```

- Directories are named as “bin”, “src”, and “mount”.
- Disk image file name will be “storage_vgc.img”.
- Place the symbolic link of the device file to your directory (In this case it is <device-file>, points to /dev/<device-file> file).
- There are four shell script files. Create those and fill according to those rules:
  - initialize.sh must create image. If image exists, it must override the previous one.

- startup.sh mounts the image and makes the virtual disk ready to run. Creates a device file for our use, attaches it to image, and creates a soft-link of that device file in our directory. If mount directory does not exist, then it creates one.
- terminate.sh unmounts the images and removes all files created by startup.sh. Detaches the device file, removes it and its symbolic link.
- purge.sh is similar to terminate.sh, also removes the created storage_vgc.img file.

To make everything clear, initialize.sh creates the first running environment, startup.sh corresponds to plugging the disk into our computer and reaching it, terminate.sh means unplugging it, and purge.sh means destroying everything.

Set up all those scripts correctly:

1. initialize.sh
2. startup.sh
   // USING YOUR IMAGE, PLAYING GAMES, ETC. //
3. terminate.sh
4. purge.sh

After executing step 3, step 2 can be run directly. After executing step 4, step 1 must be run again if you want to use your program. Purge file deletes your image, so it deletes everything you put inside that image. So be careful while running it.

4) Test Environment

The project will be run on Debian 12.8.0. Install this on your computer directly if you can. Otherwise use a virtual machine. Docker, WSL, or other software that imitates the Linux system may not work correctly.

Website: Website
Direct Image Download: Image

5) Submission

You will submit your src directory, 4 scripts described above, and your “storage_vgc.img” file with all games and main-screen executables inside the image. Create a directory with having name-surname-studentID format, place requested files inside that folder, and zip that directory.

```
<name-surname-studentID>
        ├── storage_vgc.img
        ├── src
        │   ├── src1.c
        │   ├── src2.c
        │   ├── src3.c
        │   └── main-screen.c
        ├── initialize.sh
        ├── purge.sh
        ├── terminate.sh
        └── startup.sh
```