2019/04/24

# CS3210 CONTAINER IN XV6

## 0. TEAM MEMBER

A. Yu-Ho Hsieh | yhsieh34

## 1. INTRO

In this project, we want to give a taste to implement a simplified version of container on XV6. The features include providing the basic functionalities of container and the possibility of XV6 isolation extension.

## 2. BACKGROUND

Container in real world can be understood as a standard unit of software which packs up everything needed for running an application. It isolates the execution environment from local host machine. The application running in the container should “feel” like it has its own machine. Any damage cause by the application will not affect the host machine. However, XV6 is a light weight, teaching-purpose operating system and only has basic functionalities. This limitation restricts the development of the container features but also simplifies the development process. In order to better scope the project, the developed container does not support IPC or network but only supports files isolation and execution isolation.

## 3. PROBLEMS

There are three main parts in this project. The first part is the console change. Users need a way to change the output console so that user can observe the behavior within or outside the container. The second part is the functionality of the container. In other words, user should be able to start, pause, resume and exit of a container. The third part of the project is to create an isolation of the container. The application running inside the container should not be able to break the wall and cause damage on local machine.

## 4. APPROACHES

A. Console Change

The main concept of creating different consoles and links them with different containers is to create different device files and then carefully uses **dup** and **close** system call to correctly map the stdin and stdout file descriptor. The **consoleread** and **consolewrite** function in **console.c** file need to take care of storing different input into different buffer for different device file usage. The last important part is to create a key binding interrupt from the key board so that the user can easily switch between containers and root process.

### B. Functionalities of Container

To implement the functionalities of container, we can utilize the concept of how XV6 manages processes. First is to create a struct of container to specify the fields that each container has to have such as the index of the container and on which inode it runs on. Then using a table to manage the container resource as well as their states. We can easily pick the free container when the user wants to start a new process running independently from the local machine. Through the container table, we can also pause a container by setting the program in that container with state **PAUSE**. Then we can also wake up a container by setting the program state back to **RUNNABLE**. At the end, the exit functionality is achieved by using **unlink** system call to remove the files inside a container and use **kill** system call to stop the process.

### C. Isolation

The main purpose of container is to create an isolated environment to run a process. To implement the proof of concept, we modify the function **namex** in fs.c file. Whenever the process wants to change the working directory or access the files which are not belong to its scope, the **namex** will only return the top-most inode, in other words, the inode that the container runs on.

## 5. RESULTS

The project link can be found in the reference link. The user can create a directory serving as an environment and run any program inside the environment. The delivery commands include **ctool + run / pause / resume and exit.** The commands allow user to manipulate and play around the container feature on XV6. In order to change the console output, user can hit predefined short cut (**Ctrl+T**) to observe and do experiments.

## 6. CONCLUSION

Limited by XV6, the container feature development is only a taste. However, this project demonstrates the possibility to extend the XV6 operating system to application and files isolation.

## 7. REFERENCE


A. Github: https://github.com/HsiehYuho/xv6-containerization (Under development)

B. Demo: https://youtu.be/LGeh_LAPRGQ

C. Reference Github: https://github.com/maelafifi/xv6-Containerization-OS