COL334 Assignment1

Arshdeep Singh

2016CS50625

1 Container Management

To implement the container, I created a table for container in the kernel space in the file proc.c. Table for container contains an array of container structures. Container structure maintains the uid, processes in the container and files it has permissions to see and files to which it does not have permission to. Container table is protected by a spinlock for exclusive access to it. The proc structure also has a new field, its container id, which by default is 0(the os).

1.1 Supporting System Calls

1.1.1 Create Container

Create container system call takes an integer, and creates a container with that id. This also takes care of initialising the container structure.

1.1.2 Join a Container

Joining a container involves, checking if the container already exists, and adding the process in the structure of the container. The process also gets mapped to this container.

1.1.3 Leave a Container

Leaving a container involves removing the process from the container structure which was added by the join call. The process' container id is set to zero.

1.1.4 Destroy Container

Destroying a container assumes all the processes have left the container and it just removes the container from the Container Table.

1.1.5 ps System call

When a ps is called it iterates over the process table and it prints out the process only if it belongs to the container. The container 0 can see everthing though.

2 Scheduler

The scheduler first looks for a container which has some runnable process. Container is chosen in a round robin fashion. Then the scheduler looks for the a process that can be run, that belongs to the container that is to be scheduled next. Hence fairness is ensured per every container.

3 File System

To manage the container wise virtualisation of the file system. It can be noticed every file has an inode with it and each inode has an inum which is unique to it. Thus, every container has a list of inums that it has access to. Initially, every file in the os, could be accessed by the container. Since after writing to a file, a copy of the file would be created, and the container would not have access to the original file. So, we also needed to keep track of the files that a container does not have access to. Files created by a container also are isolated from the other containers.

3.1 Create a file

If a file is created in a container then it is accessible only to the container and the inums corresponding to the files are added to the list maintained in the structure corresponding to the container.

3.2 Reading a file

While reading a file a check is done if the container has access to the file. If it has then it proceeds. No copy is created while doing this.

3.3 Writing to a file

Copy on write has been implemented and a new copy is created of a file if a file is opened with Write permissions. While doing so the older file is kept in the not allowed list of the container.

3.4 Output of ls

In output of ls, we check the accessibility of the file by the container, for this we created a system call to check if a file is accessible by a specific container. While checking if i need to list a particular file, I check three things. The file should be in my specific list of files, it should not be in my not allowed list, and if it is not in any of the specific list for any other container.