



First Project

Introduction to XV6 Operating System

Operating Systems, Fall 2021

Dr. Entezari Maleki

Amir Esmaeili - Sadra Khamoshi Far - M.Mostafa Rostamkhani

Deadline: 14 Nov. 2021, 23:59:59



Introduction

This project aims to give you a quick tour of how to work with the **xv6** operating system which has been developed at MIT for pedagogical purposes.

At the end of the project, you will have a fine understanding of the **xv6** and [system calls](#).

This guide will help you to get the **xv6** up and running. Also, it provides some sources you may want to check out to get on the right path.

Installing XV6

The following procedure has been tested on **Debian**, and should also work on Debian-based distributions such as **Ubuntu** or **Mint**. It is strongly suggested that you create an isolated workspace using tools like **VirtualBox** and a clean **Ubuntu**. You can also install it on your daily driver if you don't mind.

First, make sure to have all the dependencies.

```
sudo apt-get update && sudo apt-get install --yes build-essentials git  
qemu-system-x86
```

After successfully installing the required programs, clone the Github repository of **xv6** source code.

```
git clone https://github.com/mit-pdos/xv6-public
```

Now just compile the kernel.

```
make qemu
```



Implementation

As your assignment, you will add a new system call to xv6. This system call will return the running processes (RUNNING & RUNNABLE) in the form of an array of `proc_info` structs.

Keep in mind that the mentioned array should be sorted based on the `memsize`. If somehow you end up having two or more processes with the same memsize, just use the id (The sorting algorithm does not matter).

```
struct proc_info {  
    int pid;  
    int memsize; // in bytes  
}
```

Note

You cannot use `malloc` at the kernel level and it should only be used in user programs, so you can pass an array of `struct proc_info` with its size to the system call to fill the array.

Test your system call

Now, create a program to test the new system call. Use `malloc` & `fork` to create some new processes with different memory sizes.

Submission

Please, upload created and changed files as a ZIP file into Quera. also, attach a brief documentation on the works you have done and the problems you have faced.



Summary

1. Add a new system call, named `proc_dump` to return the running processes sorted by their sizes.
2. Create a new program to test `proc_dump`.
3. Make sure to add the test program to the `UPROGS` section in `Makefile`.

Useful Resources

- [System Call](#)
- [XV6 GitHub](#)
- [Intro to XV6](#)

Good Luck!