**Homework #2 Xv6**

Course

**CSE 460 Operating Systems**

Instructor

**Dr. Yan Zhang**

Meeting Time

**Mon. & Wed., 4:00 p.m. - 5:15 p.m.**

Due Date

**March 11, 2020**

Authors

**Kevin T. Vo**

**Esdras Lopez**

**Joseph Gonzales**

**Trevor Shortlidge**

**Brian Ayala**

# Group Members’ Information

|  |  |  |
| --- | --- | --- |
| **Name** | **Coyote ID** | **Responsibility** |
| Kevin T. Vo | 006316930 | * Implemented date system call & command (Part a) into xv6 * Implement part of uid, gid, ppid (Part c) * Implemented part of file protection (Part e) * Contributed to this report/documentation |
| Esdras Lopez | 006198864 | * Implemented part of uid, gid, ppid (Part c) * Contributed to this report/documentation |
| Joseph Gonzales | 006242648 | * Implemented ps command and Ctrl+P (Part d) * Contributed to this report/documentation |
| Trevor Shortlidge | 006310209 | * Implemented part of Ctrl+P (Part b) * Implemented part of file protection (Part e) * Contributed to this report/documentation |
| Brian Ayala | 006191688 | * Implemented part of Ctrl+P (Part b) * Contributed to this report/documentation |

# Table of Contents

1. Demonstration of the date() System Call ……………………………………………4
2. Demonstration of the Ctrl-P Special Control Sequence…………………………
3. Demonstration of UIDs, GIDs, and PPIDs……………………………………….
4. Demonstration of the “ps” Command………………………………….
5. Demonstration of the File System Protection Features……………………..

# **Demonstration of the date() System Call**

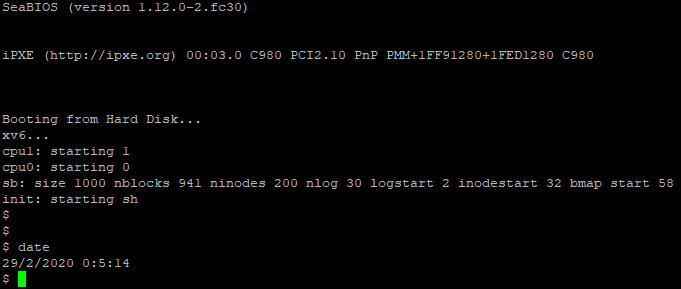
The system call “date” has been implemented to the Xv6 operating system where it would display the current day, month, year, hour, minute, and second in the format below:

*DAY/MONTH/YEAR HOUR:MINUTE:SECOND*

During the addition of this system call the following files have been created/modified:

|  |  |  |  |
| --- | --- | --- | --- |
| **Filename** | **Created**  **/ Modified** | **Line Numbers** | **Code** |
| syscall.h | Modified | 23 |  |
| user.h | Modified | 26 |  |
| sysproc.c | Modified | 95-106 |  |
| usys.S | Modified | 32 |  |
| syscall.c | Modified | 106  &  136 |  |
| date.c | Created | 1-18 |  |

### Result from Execution of date() System Call:



# **Demonstration of the Ctrl-P Special Control Sequence**

1. **Demonstration of** **UIDs, GIDs, and PPIDs**

The implements the feature of users and groups into xv6 through uid and gid where it be used to store ID unsigned integers for tracking the ownership of a process for a user or group. By typing “id” into the terminal when executing xv6, the system call will invoke and demonstrate this feature by displaying the UID, GID and PPID that has been established by the test function within “id.c”.

|  |  |  |  |
| --- | --- | --- | --- |
| **Filename** | **Created**  **/ Modified** | **Line Numbers** | **Image** |
| syscall.h | Modified | 25-29 |  |
| user.h | Modified | 28-32 |  |
| sysproc.c | Modified | 114-162 |  |
| usys.S | Modified | 34-38 |  |
| syscall.c | Modified | 108-112  &  138-142 |  |
| id.c | Created | 1-26 |  |

Description of changes made within the file “syscall.h”:

* The system calls responsible for setting and returning the UID, GID, and returning PPID are added so that it system call name can be mapped to a system call number so that it can be invoked within xv6.

Description of changes within the file “user.h”:

* The prototype to the getter and setter functions for setting and returning the unsigned integers of UID and GID including one getter function for the PPID.
* The prototypes included are to the functions that holds the code to perform the task that a user uses a system call to request for.

Description of changes within the file “sysproc.c”:

* sysproc.c contains the functions:

uint getuid (void) // UID of the current process

uint getgid (void) // GID of the current process

uint getppid ( void) // process ID of the parent process

int setuid (uint) // set UID

int setgid (uint) // set GID

to actually sets the UID and GID using the setuid(uint) and setgid(uint) functions.

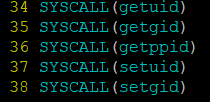
Note that user/xv6 will use these two functions to set the UID and GID.

The getter functions uint getuid(void), uint getgid(void), and uint getppid(void) that returns the UID, GID, and PPID respectively.

It is within setter functions where it checks if the value UID and GID are within range of [0, 32727].

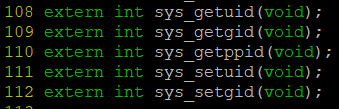
Description of changes within the file “usys.S”:

* “usys.S” contains the system calls to be made available by the kernel:



Description of changes within the file “syscall.c”:

* First contains the system call entry points:



* Secondly, “syscall.c” contains a dispatch table that defines from the symbol name made earlier within files “usys.S” and “syscall.h” to its function name. All of this is done within the “int (\*syscalls[])(void)” within the “syscall.c”.

Description of changes made within file “id.c”:

* Contains the test function to run the system calls implemented in this part.

### Result from UIDs, GIDs, and PPIDs:

1. **Demonstration of the “ps” Command**
2. **Demonstration of the File System Protection Features**

The task requires you to implement a completely new abstraction in xv6: file system protection. This is a very large area, so you will focus on a small subset of protection that will nevertheless give you insights into how to implement additional protection and security concepts. You will implement file system protection, user programs for manipulating those protections, and add protection checking to the exec() system call.