CS 533 Final Project: eXpOS Summary

Zhan Li
Haomin He
Yifeng Liang

Introduction

eXpOS(eXperimental Operating System)[1] is a online teaching system which helps college students to learn the concepts and implementation of essential OS components. Following the tutorial, the students would be able to learn the implementation of important OS data structures and kernel routines by completing a series of programming assignments. The project assumes that the student has undergone a course in computer organization, and is comfortable with programming.

According to the project documentation, eXpOS is designed to be run on the XSM (eXperimental String Machine) architecture which is a simulated machine hardware. XSM is an interrupt driven uniprocessor machine which handles data as strings. Upon installation, users would be able to start with a simulated bare machine with no software on it. Following the roadmap, students can get familiar with various mechanisms to write the kernel modules and get them as XSM executable programs.

To finish the project roadmap, students must get familiar with the following mechanism or knowledge:
1. Architecture of XSM and essential components.
2. SPL(System Programmer's language)
3. EXPL(language for user application)
4. XFS interface(The interface which allows user to communicate with XSM's storage).

As we mentioned above, students start with a bare machine with no software on it. The assignments let students to write OS kernel code using SPL language. The compiler will build user code into assembly instructions and stores them as a part of the system modules. Upon finish of any of the steps in the roadmap, students can program user applications using EXPL language to test the existing functionalities.

Roadmap

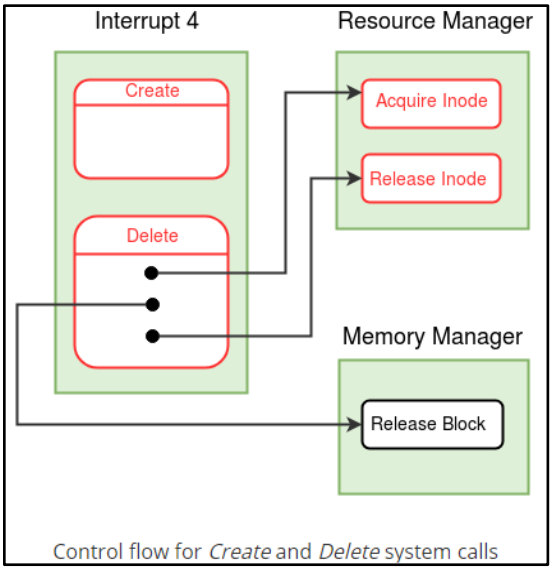
eXpOS provides a step by step guidance towards writing a small operating system from scratch. This roadmap is divided into 27 stages, which students can finish stages in sequential order and eXpOS will be built incrementally. Every stage requires students to read documentation and implement the required OS components.

Here we introduce one of the stage that we found interesting, stage 23 file creation and deletion[1]. There are three file system calls in eXpOS:
1. Create system call creates an empty file and initializes the new file metadata in inode table and root file. Inode table and root file stores details of every eXpOS file stored in the disk. The user executing the Create system call becomes the owner of the file.

2. Delete system call deletes the record of the file and releases the disk blocks. A file can not be deleted if it is currently opened by one or more processes. Delete first acquires the lock on the file by invoking Acquire Inode function (waits until the file becomes free, checks whether the file is deleted from the system because other process may delete the file during waiting period, then locks the file); Delete invalidates the record of the file in the inode table and root file; Delete releases the lock by calling Release Inode function.

3. Shutdown system call commits the changes made by Create and Delete system calls in the memory copy of the disk data structures back into the disk. This invokes the Disk Store function, which takes a process, a page number and a block number as arguments, updates the disk status table, and initiates the store operation for given page number and block number. When store operation is completed, system raises the disk interrupt which makes this process ready again.

The figure below is a control flow for Create and Delete system calls:



Control flow for *Create* and *Delete* system calls

Comparison with other Teaching OS

1.Documentation Reading

Blitz has very detailed step by step documentation about implementation in projects, which help immensely. Documentation of eXpOS is well-organized and in a good order for students to read

through, however, the content of the instructions in the complex task is not very specific. A beginner may need extra documentation for implementation, as the complexity may confuse students.

2.Platform Setting up

Blitz, eXpOS and xv6 are compatible with the popular OSes today. The difference is the instructions for setting up in different Operating system. Blitz has step-by-step instructions for MacOS, windows and Linux[2], although it may take a while for students to configure. Students should be able to make it by themselves based on the available documents from the official documents. Xv6 is a teaching operating system developed in the summer of 2006 for MIT's operating systems course[3]. Students can find how to set up from the official websites. Linux is recommended as the platform for the tutorial. eXpOS needs several software packages. The documentation only covers instructions for Ubuntu and Debian[4]. For other platform like MacOs, students may need to configure packages for themselves.

3.Workload for Students in a Quarter

Blitz system is developed by Harry H. Porter III, Ph.D for supporting undergraduate level course on Operating System[6]. Students needs to complete 5 projects independently, which fits perfectly within PSU's short school term. Xv6 is designed for an entire semester[5], which we don't think is possible for PSU students to accomplish all of the tasks within two months. On the other hand, eXpOS is designed for a quick term, but we found that the learning curve is a bit steep.

4.Grading Policy

Grading for blitz could be based on the small projects, while eXpOS has 27 stages, which would require a much more extensive and complicated grading policy. With so many stages, dividing the grade up equally poses a challenge as many of the stages are closely related to each other.

5.Miscellaneous

NACHOS is different from eXpOS. NACHOS learning system allows application programs in the noff(Nachos Object File Format) executable format, running on a software simulation of a MIPS machine, to invoke system calls. However, the OS code users write doesn't run on the MIPS machine. The code is actually C code running on the Linux/Unix machine. When an application program invokes an OS system call, the MIPS simulator transfers control to a corresponding "stub" function in the simulated environment. Users must write C code in the stub to do "whatever is expected from the OS" to satisfy the calling application program invoking the system call. Since the MIPS machine is simulated, user code has access to its memory, registers etc. Thus users can implement the system call, put return values in appropriate memory locations on the simulated MIPS machine and transfer control back to the calling program[7].

Accomplishment

Everyone completes stages 1-12. Then we work on remaining stages together and do research on different educational operating systems. In our opinion, Blitz system is suitable for undergraduate-level courses. Compare to Blitz, eXpOS may be challenging for undergraduate students. Adding more detailed instructions may help users understand better about the system and implement tasks. xv6(MIT version) should be modified to satisfy PSU short term.

References

[1] http://exposnitc.github.io

[2] http://web.cecs.pdx.edu/~harry/Blitz/OSProject/p1/proj1.pdf step3

[3] https://pdos.csail.mit.edu/6.828/2016/tools.html MIT instructions

[4] http://exposnitc.github.io/faq.html question 8 and answer

[5] https://en.wikipedia.org/wiki/Xv6#Purpose

[6] http://web.cecs.pdx.edu/~harry/Blitz/index.html overview part

[7] http://exposnitc.github.io/faq.html