XOS : Experimental Operating System

Shamil C. M.
National Institue of Technology
Calicut, India
shamil_bcs09@nitc.ac.in

Sreeraj S
National Institue of Technology
Calicut, India
sreeraj_bcs09@nitc.ac.in

Vivek Anand T. Kallampally
National Institue of Technology
Calicut, India
vivekanand_bcs09@nitc.ac.in

K. Murali Krishnan
National Institue of Technology
Calicut, India
kmurali@nitc.ac.in

ABSTRACT

In this paper, we introduce you to an operating system project that helps undergraduate computer science students learn and understand the basics of building an operating system. The specification of XOS or Experimental Operating System has been laid out for students to build it from scratch in a bottom-up manner. XOS runs on a simulated machine hardware with a simplified innate instruction set and utilizes its own filesystem. Unlike other common instructional operating systems, the complete development environment including custom programming languages, debugger, file system interface and a highly instructive roadmap for sequentially building XOS is provided. A student building XOS will implement features like multiprogramming, file systems, process management and virtual memory management.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms

Design, Experimentation

Keywords

XOS, instructional operating system, experimental operating system

1. INTRODUCTION

Teaching operating systems has been a challenge at undergraduate level. To tackle this problem several instructional operating systems like Nachos[2], OS/161[3], Pintos[4], GeekOS[1] etc. have been developed by various universities. Nachos[2] has been one of the most popular of the instructional operating systems available and used in many institutes across the world [1]. Although Nachos implementation is fairly simple, it uses a mixed mode approach, where the operating system kernel is co-resident with the machine simulator and fused together as a single program, and does not provide an intuitive idea of the separation between these components. Moreover Nachos[2] and OS/161[3] runs on top of MIPS machine simulator, host machines running on other platforms require cross compilers and separate tools. The authors of OS/161 has expressed plans of moving towards a different architecture due to lack of freely available tools [3]. XOS or Experimental Operating System, which we propose as a project for undergraduate operating systems laboratory courses, addresses these issues, by providing an original mahcine architecture known as XSM (Experimental String Machine) which has its own instruction set. This completely new and easy-to-understand instruction set helps avoid complexity associated with actual architecture and helps students to focus on the operating system aspects. The complete package including the high level languages for application as well as system programs, their cross-compilers to XSM machine architecture, simulator and debugger for XSM is provided for building XOS from scratch. In XOS, there is a clear differentiation between the machine and the operating system kernel like OS/161[3], which is more realistic towards the actual scenario and has its own set of tools.

Instructional operating systems like Minix and Xinu [1], provide a functional operating system on which modifications are to be done by students. Almost every other instructional operating system provides a skeleton of an operating system. However in XOS, only the specification has been laid out, and students learn to implement XOS from ground up using the tools provided. Most instructional operating systems use C/C++ or Java for programming. Knowledge of a particular high level language becomes necessary for programming the operating system. In this project, a simple high level language called APL (Application Programmer's Language) and its cross-compiler to XSM instruction set is provided to write user programs to test XOS, and an XSM machine dependant language called SPL(System Programmers Language and its cross-compiler is provided to program the OS itself. Most instructional operating systems use the UNIX filesystem for file management by the operating system. XOS is different from other instructional operating systems by providing its natve file system known as XFS (Experimental File System). An interface to between the UNIX filesystem and XFS filesystem is also provided.

XOS has features like multiprogramming, process management, filesystem and virtual memory. A sequence of stages are provided in an instructive roadmap which helps students to build XOS sequentially. Although certain simplifications have been made in XOS, compared to real systems like absence of blocking system calls, device management and file caching, the elementary and fundamental aspects of operating systems and data structures have been retained. Process synchronization have been completely left out because, the it will turn out to be too much overwhelming for a student in a 16 week semester. The further sections describe in detail the various components of XOS.

2. SYSTEM COMPONENTS

As explained in the previous section, an the primary components of the project include a simulated machine hardware (XSM), file system (XFS) and the operating system itself (XFS). Apart from the primary components, various tools have been provided as part of the development environement. They include languages like APL and SPL and their cross compilers to XSM instruction set is also provided, XSM debugger, and a UNIX-XFS interface to transfer files between a UNIX machine and XFS disk. The XFS disk is basically a file in UNIX machine which acts as a disk for XOS.

2.1 Experimental File System (XFS)

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[1] typeface, but that is handled by the document class file. Take care with the use of[2] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*.

2.2 Experimental String Machine (XSM)

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

2.2.1 *dasa*

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin. . .\end` construction or with the short form `$. . .$`. You can use any of the symbols and structures, from α to ω, available in LaTeX[?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}x=0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

[1]A third footnote, here. Let's make this a rather short one to see how it looks.
[2]A fourth, and last, footnote.

2.2.2 *Display Equations*

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}x=0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty}x_i=\int_{0}^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

2.3 Experimental Operating System (XOS)

Citations to articles [?], conference proceedings [?] or books [?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item
d in the proper location in the `.tex` file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[?].

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

2.4 Development Tools

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

2.4.1 *Application Programmer's Language (APL)*

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.4.2 *System Programmer's Language (SPL)*

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.4.3 *XSM Debugger*

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.4.4 *XFS Interface*

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location

Figure 1: A sample black and white graphic (.ps format) that has been resized with the `psfig` command.

deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

3. ROADMAP

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these

4. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

5. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

6. REFERENCES

[1] C. L. Anderson and M. Nguyen. A survey of contemporary instructional operating systems for use in undergraduate courses. *Journal of Computing Sciences in Colleges*, 21(1):183–190, 2005.

[2] W. A. Christopher, S. J. Procter, and T. E. Anderson. The nachos instructional operating system. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, pages 4–4. USENIX Association, 1993.

[3] D. A. Holland, A. T. Lim, and M. I. Seltzer. A new instructional operating system. *ACM SIGCSE Bulletin*, 34(1):111–115, 2002.

[4] B. Pfaff, A. Romano, and G. Back. The pintos instructional operating system kernel. In *ACM SIGCSE Bulletin*, volume 41, pages 453–457. ACM, 2009.

Table 2: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables