Using Inferno for an Advanced Operating Systems Course

Francisco J. Ballesteros
Sergio Arevalo
{nemo,sarevalo}@gsyc.inf.uc3m.es

January 12, 2001

Abstract

During spring 2000 we faced the question of how to design an Advanced Operating Systems Course at Universidad Rey Juan Carlos of Madrid. After browsing through the internet, we found that on most occasions, a not-so-advanced operating system was being usually used to teach such kind of course.

This brief note describes the experience of using an state-of-the-art Operating System which includes advanced concepts to teach a second-level course on Operating Systems.

1 Introduction

It is hard to teach operating systems because, in this field, theory and practice mix probably more than in any other field. To make an example, it is feasible (although not advisable!) to teach theory of compiler construction without making the students build a compiler. Of course, giving the size of a simple compiler, it is good practice to organize a course so that students can learn compiler theory and then build their own compiler. However, considering operating systems now, most concepts in the field are of interest only if they make the resulting system perform well. This means that practice is so related to theory that students are not likely to understand theory unless exposed to a real system. Besides, to make things worse, it is not feasible within a single course to assign the construction of an operating system as a student project.

At Universidad Rey Juan Carlos we have three courses on Operating Systems. The first two ones are part of a three years first cycle of Computer Science Engineering. The last one is part of a two years second cycle of the same career. During the second course on (Advanced) Operating Systems (AOS), at third year, students learn about using and building advanced operating systems for use in our networked world.

In the past few years, AOS was using unix (Linux) as the operating system used for labs and assignments. Students learned how to use TCP and UDP using

a wrapper for sockets by building a chat application. Although this experience proved fine in that students could learn how to apply concepts of distributed systems theory to build a distributed application, we found that students were mostly unable to use sockets due to its complex interface as well as they were unable to think of how to structure a distributed system in a way different from the traditional one: unix-plus-sockets.

Although one alternative could have been using the real socket interface for this semester (so that at least students could learn how to use them), we thought that it was most convenient to open their minds and let them see how a novel way of structuring a distributed operating system could work. The aim is to prepare them for building distributed applications and systems in the most convenient way.

This semester students have been using Inferno [2] and Limbo [5] to build a distributed mail service upon the Styx architecture [4]. By doing so, they have seen how a distributed service can be structured in a simple way by inventing abstractions well-suited for the application at hand. In particular, they were surprised that by mapping parts of mail messages to files, and using a common protocol (styx) to access files, their application could be built easily. Although we do not know, we suspect that in the past they would have built something very different (and much more complex) to implement the same application.

In what follows we try to describe my perception of this experience. Although we feel it succeeded, it is really too early to know and what is said should be considered to be just my opinion. we hope it is useful any way.

2 AOS with Linux and sockets

The first time students are exposed to operating systems is during their second year Operating Systems (OS) course. This is a traditional course where they learn basic OS concepts following a classical textbook [6]. This includes concepts about system structure, process management, memory management, file systems, and I/O. Assignments focus on using the shell as well as on using system calls (e.g. build a shell).

Although the OS course has plenty of examples, many of which come from Plan 9 and Inferno, after the OS course (before the AOS one), students have used a centralized system and are not yet prepared for a networked world. At this point they arrive to AOS.

The AOS course is devoted to teaching advanced operating systems concepts. It revisits operating systems concepts, considering this time the network as a central element. This means that the course is actually a mixture of a distributed systems course and an operating systems course. Theory lectures follow a classical textbook [7] (which we are going to change soon, by the way). It includes concepts about structuring a distributed system, as well as communication, synchronization, process management and file systems concepts for distributed systems.

During last years, AOS featured assignments with sockets and Linux. Actu-

ally, students did not use sockets, but a wrapper for them instead—to simplify their interface. Students built a chat application to learn distributed systems concepts. Although the result was quite good, they were using mostly a centralized system together with a messaging mechanism. This means that after the instructor taught them that "unique identifiers are important in a distributed system", they found later by themselves that "identifiers supplied by the *real* operating system being used only have sense within a single node". We would prefer the assignments to re-enforce the concepts rather than to fight with them.

Being AOS an operating system course, it would be desirable to let students build their own system. Admittedly this is not feasible due to the size of the problem. The next best approximation would be to let students extend a system already built. This is not feasible either with UNIX because the system (Linux) is so complex that students would have problems doing so—not to talk about the centralized design of the system. Even the next desirable approximation (just using a distributed system) is infeasible with UNIX, because it is not a distributed system.

At least, students should be able to see how a decent system works, how to use it, and how to extend it. To fix this problem, we have introduced Inferno as the operating system of choice for AOS.

3 AOS with Inferno

For this semester, students have been using Inferno

For the first time, they have been using a system built along the concepts taught in theory, and not a centralized system. This is not to be underestimated. To continue with the example used above, this time students have found that things like QIDs and FIDs are important and follow the concepts taught in theory lectures. And this happens as a "side-effect" of the assignment, which is not centered on that concept! Of course, we have been using just a silly and simple example, but this happen with many different concepts.

One good thing of using Inferno for the course is that students can feel how they extend a real system. Since Inferno is about files serviced through Styx channels, it is affordable for a student to extend the system by supplying a given service through Styx. This year, the service chosen was a distributed mail service. More information about the course can be found in a (spanish) web page [1].

An important consequence of the Styx assignment, is that the student can learn how implementing a system can be greatly simplified by choosing an appropriate abstraction—which is mostly what AOS is about. The assignment is described in an informal way, so that the student has to think about how to map the service to the abstraction. It is of help that networking in Inferno is really easy for the programmer. It takes several orders of magnitude less time to teach how to use connections in Limbo than it takes to teach how to use sockets on UNIX. Most of the time can be therefore devoted to learning the concept considered, and not the mechanics of the system.

It could be said that implementing a Styx service is not preparing the student for the "real world"; but we think this is not the case. My students were really motivated by the fact that what they are doing as assignment is really similar to what they will have to do in the industry when they start to implement applications in the Internet. All in all, applications in the web are mostly servicing files through a given protocols; most of the work to build an application in this domain is to map the service into the abstraction. We sincerely think that the reader would clearly perceive the isomorphism.

Considering now more practical aspects of the organization of the course, Inferno has been of help here too. In previous years, the student would use the Linux laboratory where several dozens of PCs are running Linux. This is a physical room, with student home files serviced through NFS from a central server. This year, the AOS laboratory still uses the Linux laboratory room, but it has become virtual! Student home files are serviced through Styx from an Inferno server running on the Linux file server. Given the portability of the system, this means that almost any PC in the University can be considered part of the Inferno laboratory: it has become distributed!

I have found that besides the PCs in the Linux room, some PCs from research projects (used by students) as well as some home PCs (linked through slow modem lines) have been used as part of this "Inferno laboratory".

A funny incident in this respect is that initially, we planned to use a Windows NT room for the Inferno laboratory. Due to problems "introduced" by the system administrators, it was infeasible to do so. Amazingly enough, this was not even a bit of a problem, because Inferno was ready to run on another place—a Linux room.

Moreover, even though students are advised to install Linux or Plan 9 at home as their base system, they no longer *must* do so. Some of them are using Linux, some of them are using Windows; yet all of them can run the assignment on the same system, Inferno.

Regarding the usage of the system at home, it would be better if the system could be "open source"; and we have tried to choose open source systems to teach our courses. Although a source license can be purchased quite cheaply by students, it is still expensive enough for Spanish standards. In any case, the binary-only Inferno distribution, distributed for free by Vitanuova suffices for our purposes[1].

To avoid misunderstandings regarding the system source, we have to say that the license allows students to access temporarily the source code in the system installation made for the course. One of my students made a minor fix to the keyboard driver, to allow the use of a modifier key in X windows to type characters accessed with "AltGr" in the Spanish keyboard. Although the fix was not good enough to be distributed to others, this proves that the student still can learn how the system works and how to enhance it.

Another beneficial "side-effect" of using Inferno has been using Acme as

[1]For the "Operating Systems Design" course following the AOS one, we are using Plan 9, which is distributed open source.

the developing environment [3]. Most students came out of our courses using XEmacs, Vi, the shell, and typical graphical user interfaces of Unix today. Acme has shown them how a simple yet effective environment can be built with a small fraction of the resources used by the more traditional environment. Whenever a student was surprised about this and asked me, we suggested him to compare the memory footprints of the applications involved, and also to compare the apparent speed of several Inferno applications with similar ones written in Java. We think they learned a lesson there.

4 Lessons learned

To summarize, we think students learn more by using a system built along the lines taught to them during theory lectures. Using a hosted environment is important in that the location of the labs is no longer relevant, and in that the environment perceived by the student is always the same one. By using Styx, they are learning a novel way of structuring a system while at the same time they are learning how to build applications for the internet in today computing platforms. By using something different from UNIX/Windows, they learn that there are better ways to build Computing Systems.

As a suggestion, we miss a tutorial on using Styx to build a file server. We may write one for the next semester, if none is available by that time.

In few words, although AOS is still unable to let students build their own system, they are using and extending a distributed system for their first time.

References

[1] Francisco J. Ballesteros. Advanced Operating Systems Course Web site. http://gsyc.escet.urjc.es/docencia/asignaturas/ampliacion˙ssoo, 2000.

[2] Sean Dorward, Rob Pike, David Leo Presotto, Dennis M. Ritchie, Howard Trickey, and Phil Winterbottom. The Inferno Operating System. *Bell Labs Technical Journal*, 2(1), 1997. Also in http://www.vitanuova.com/inferno.

[3] Rob Pike. Acme: A User Interface for Programmers. *Plan 9 Programmer's manual, 3rd ed.*, vol. 2, 2000.

[4] Rob Pike and Dennis M. Ritchie. The Styx Architecture for Distributed Systems. *Bell Labs Technical Journal*, 4(2), 1999. Also in http://www.vitanuova.com/inferno.

[5] Dennis M. Ritchie. The Limbo Programming Language. http://www.vitanuova.com/inferno/papers/limbo.html, 1997.

[6] Andrew Tanenbaum. *Modern Operating Systems*. Prentice-Hall, 1992.

[7] Andrew S. Tanenbaum. *Distributed Operating Systems*. Prentice-Hall, 1995.