A Nifty OS Project for Today's Student

James S. Jones
Department of Computer Science and Information Technology
Graceland University
1 University Place, Lamoni, Iowa 50140
jsjones@graceland.edu

Abstract

Operating Systems (OS) covers the internal structures and mechanisms used to manage a system's resources. It is a challenge motivating today's application-driven student to fully comprehend the finer points regarding process states, suspension, burst times, interrupts, parent/child tasks, multi-threading, semaphores, locality, page faults, working set, and more. System programming or simulation projects are daunting for the computer information systems (CIS) student and evade the larger picture for the computer science (CS) student. I describe a comprehensive OS project that engages both CS and CIS students and covers a wide range of fundamental concepts introduced in the course. Student teams must find, install, and experiment with utilities that illustrate key concepts, develop a technical guide with illustrations, and give a presentation/demonstration to the class. This project taps into the inclination for today's student to seek, download and play with unfamiliar software and it promotes discovery, analysis, documentation, oral presentation, and teamwork.

Introduction

For some time I have thought the Windows Task Manager utility would be a great tool for illustrating fundamental Operating Systems concepts[1][2] on process-, memory-, and I/O-resource management, if I only took the time to fully explore it. I also knew that third party utilities existed that might be even better for observing and demonstrating such system internals. Lacking the time to do it myself, it occurred to me that students today are so adept at finding, downloading, installing, and playing with all kinds of software that this would be a great project to hand over to them and see what they could do with it. So that is the kernel thought from which the following project emerged.

In fall 2010, I taught Graceland University's Operating Systems and Networks course, an upper-division requirement for our CSIT majors whose interests range from computer science to information technology. By midterm I had covered most of the classic OS topics associated with process-, memory-, and I/O-management and wanted to assign an OS project for teams work on in the second half of the semester, concurrently with lectures and exercises that shifted to networking concepts. The *Project Handout* section that follows is what was given to the students. This is the only material needed by other faculty who want to try a similar assignment. In short, each team is asked to find software tools that can demonstrate the OS concepts that we had covered, become familiar with it through experimentation and online helps, create their own illustrative and descriptive guide for OS techies on how to use it, and give a presentation and demonstration to the class. Everyone in the class got to have a copy of the how-to guides and power point slides created by all of the teams.

Our department has a CS lab with 18 workstations on our university's domain, all being quad-processor PCs with Windows 7. I was able to log on as local administrator to install software and offered to do so for any team who asked for it. The utility chosen by most teams could be run from a flash drive without installation so they did not make that request. A bigger issue was the need for administrative access in order to fully use some features and control system processes. To resolve this issue, teams were given privileged access to older spare workstations. The teams also used laptops owned by their members since that was convenient and gave them administrative access. A beneficial side-effect of all this was that students saw variations due to different processors (quad-, dual-, or single-processor), different privileged status, different network conditions, and different systems (Vista, XP, or Win7).

Not knowing what to expect for the first time assigning this project, I gave students the freedom to choose whatever operating system platform they wanted and whatever utilities they could find as long as it was cost free (even if only for a trial period). Teams had to identify their choices early on and submit draft documents midway so that I could guide their efforts and avoid total procrastination. These were oral instructions that were not included in the written specifications, as given in following *Project Handout* section.

Project Handout (addressed to the student)

It is always a bit of a mystery to figure out what is going on with the tasks and services running on a PC. Internally, the OS is keeping data structures and metrics to manage processes, memory, and devices. There must be tools to visualize these details and help us understand the internal details and activities. You are to find such a software tool or tools that do this (built-in utilities and/or free software). You are expected to focus on details that an OS enthusiast would want to know (OS principles and internals learned in this class). You'll create a user document, complete with table of contents, section titles, installation instructions (if applicable), easy to read text, graphical illustrations, and screen shots. Create your own screen shots and descriptive text (nothing copied from existing guides). Your team will present this in class (power point slides) and demonstrate it if possible. Your user guide and your power point slides will be submitted to the instructor and made available to the entire class before the presentation.

To help you think this through, here are some things an OS enthusiast might want to know:

- What percent of time is a task in each of the Run, Ready, and Wait/Blocked states
- What measures are displayed to determine how I/O-bound or CPU-bound a task is
- What is the average time or percent of time a given task sat waiting for I/O
- What is the average CPU burst (i.e., Run time between Blocked states) for a task
- How many times did a given task make a request for more memory
- How many network connections (sockets) did a task open or close
- Is there a synchronization (semaphores or mutex primitives) between certain tasks
- How many threads does a particular task have at the moment
- What tasks are in some sort of suspended state (swapped out to disk)
- What I/O measures exist that describe disk, flash drive, and other device activities
- What's the page size and how many pages of virtual memory does each task have
- What information can we see regarding a task's pages stored in real memory
- What is the working set size for a task (or average working set size for all tasks)
- Are there global or task specific statistics for page hit-to-fault ratios

You may not be able to answer all of these questions, but perhaps you can answer some variant of them—if not a measurement on a specific task, then perhaps a total, average, or indirect measure for all tasks. If the software gives charts and graphs, those are great for screen shots but make sure to illustrate to the user how to pull out some specific details or information for them. When you create screen shots, you should crop them in a useful way and reduce their size (click and drag the corner of the image pasted into the doc) so as not to take up too much room. Make it look professional, with single spaced paragraphs and 1" margins on all sides. Give captions with figure numbers for illustrations and screen shots. Write descriptive text that point out various things in any given illustration. Screen shots that have only one or two sentences between them are not what I am looking for in this guide. I want your text to be somewhat instructional too, so that the reader will encounter brief descriptive reminders on the meaning of key terms to go along with the how-to instructions that involve them.

Tool Selections

Although the teams were told that built-in utilities like the *Windows Task Manager* [3] were acceptable, all teams chose to find and download other utilities. This suggests that today's students are motivated to do their own searches and try out unfamiliar software. The tools chosen by the project teams displayed similar data as the *Task Manager* but had more control features. Figure 1 is a sample screen shot of processes ordered by memory allocation and figure 2 shows the range of other data one can select from the View menu.

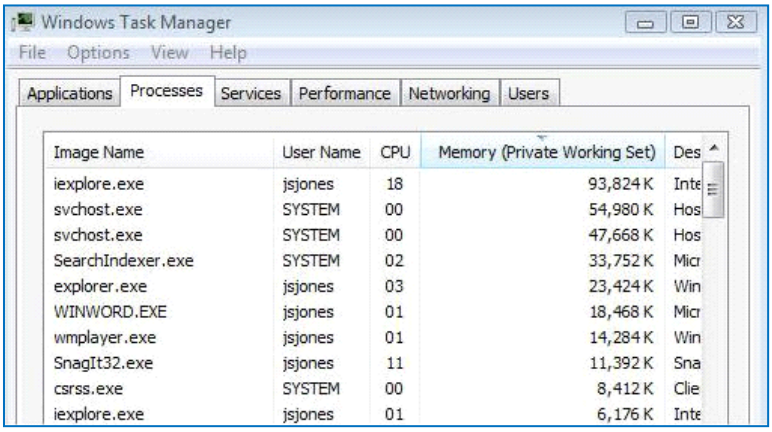

Figure 1: List of processes in Windows Task Manager

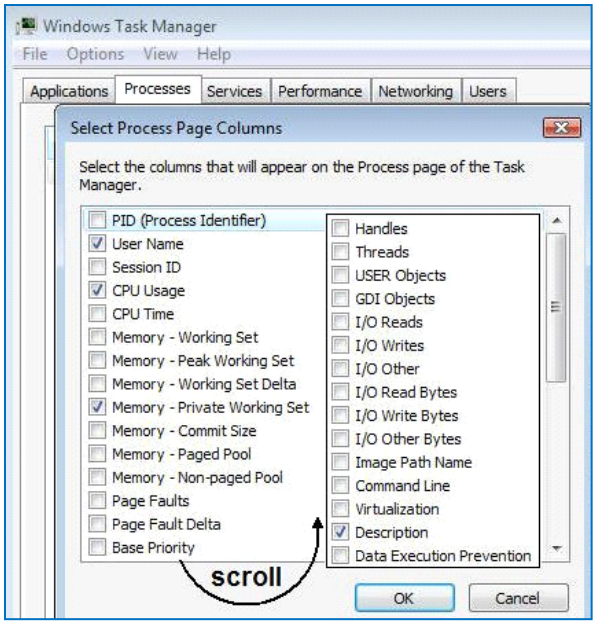

Figure 2: Process specific data to choose from

Teams whose members had Macintosh proficiency were encouraged to use that platform, but no team did. The most likely reason was the prevalence of Windows systems on our campus and among fellow students. A Macintosh team could have used the built-in *Activity Monitor* [4] which is similar to the *Windows Task Manager*.

One team used *WhatsRunning* [6], a third party utility that employed the Windows Management Instrumentation (WMI) subsystem and they needed me to install it on a lab machine as Administrator. Figure 3 shows a screen shot from that team's document in a section describing working set size metrics (shared, private, min, max, and default) and how to change them. Teams were expected to make their own screen shots and add markings for clarity to accompany their step-by-step instructions, as was done in figure 3.

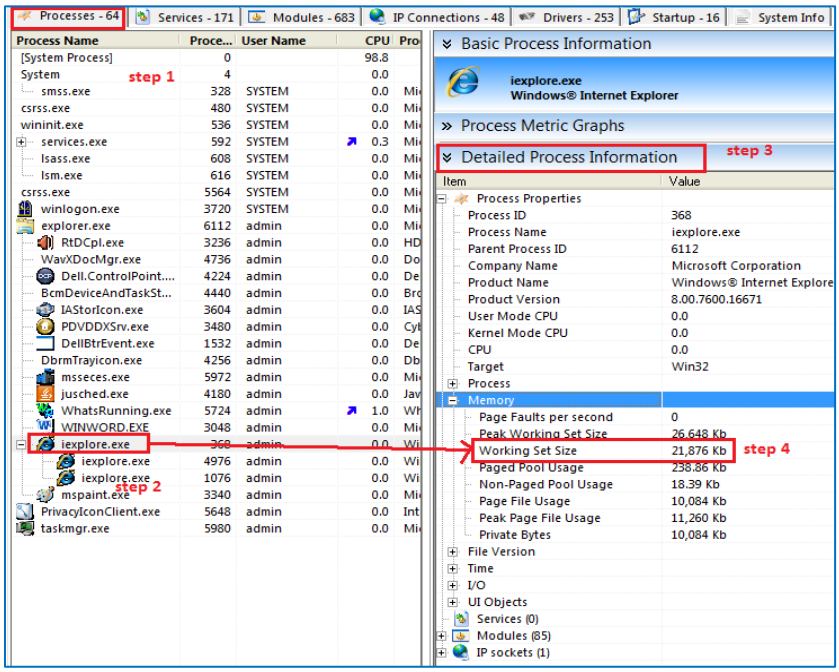

Figure 3: WhatsRunning

All other teams used *Process Explorer* [5], a freeware utility available from Microsoft. It is easy to use since it can be run from a flash drive without administrator privileges. A clip of the opening screen is illustrated in figure 4 from which a user has many choices.

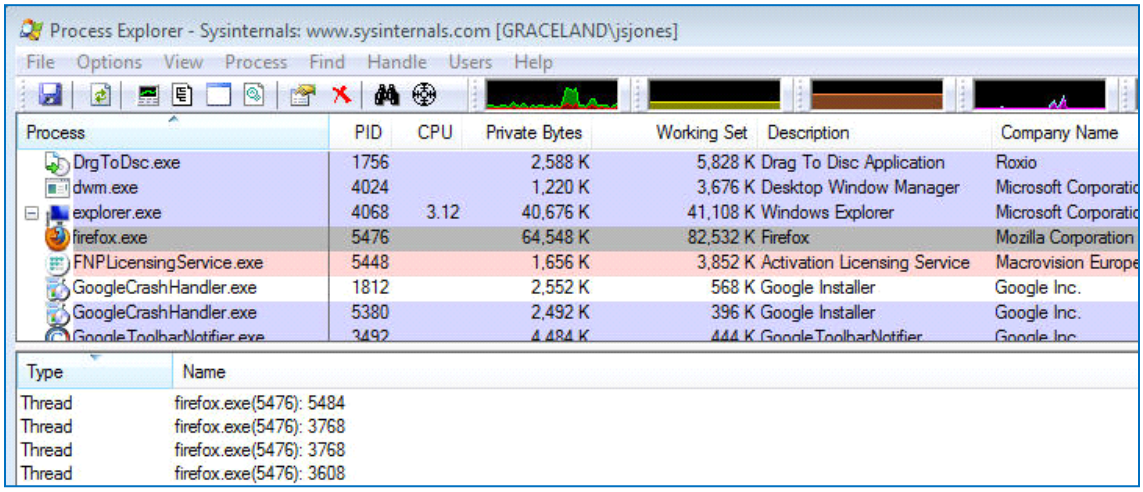

Figure 4: Process Explorer

By right-clicking a particular process the students could access its properties and make changes that effected how the system treated it or its threads. There are far too many interesting and relevant screens to attempt to show here and the figures in this paper are not intended as a guide. It is enough to say that the students had a wealth of things they could see and do to make their own discoveries.

Outcome

I could not have been happier with the results. Students who were starting to lose interest or become overwhelmed by the fundamental concepts covered earlier now became fully engaged and energized by this project. Their experiments were what interested them and they wanted to share any quirky or unexplained behaviors with fellow classmates and the teacher. For example, one student discovered how to bring the system to its knees by launching many instances of the calculator by repeatedly pressing the special calculator button on the keyboard. His team used Process Explorer to try to better understand what was going on. Another favorite activity was to kill the parent task to see what happened to the child tasks or vice versa.

It turned out to be important to have teams turn in a draft of their work midway through the project since that is when the ongoing demonstrations and questions really took off. Furthermore, it allowed me to give critical feedback to improve the quality of the final documents. With every team I had to ask for more specificity and elaboration in their writing, which I got in the final product. I also pointed out where graphical illustrations could be reduced in size or made clearer. The draft submissions were full of oversized images with bullet point outlines or short sentences sandwiched between them, however the final products were all quite good.

The group presentations were also well done. The only glitch was that one team followed another team whose work was based on the same software utility, and they floundered somewhat because some of their information had already been presented to the class.

An interesting misconception arose which led to some great discussion when a team identified a task being CPU bound because it lacked recent read/write activity, not considering the possibility that it was an interactive task simply waiting for user input.

Certain processes emerged as favorites for learning, such as Notepad, web browsers, the monitoring utility itself, and the system (i.e., process ID zero). These provided a range of things to observe with single-threaded, multi-threaded, and process-spawning tasks.

Possible Improvements

Instead of having students find the utility of their choice, Process Explorer could be specified and made available on lab PCs. This puts the focus on analysis and also lends itself to smaller exercises that could be specific to each OS concept when it is first taught.

The current *Project Handout* is only suggestive in its list of questions that might be addressed. One could be more definitive in the specifications and publish a grading rubric that is tied to specific concepts that must be demonstrated. However, I was pleased with the outcome as it stands because I took the opportunity to make definitive suggestions when the drafts were submitted and considered the extent to which they were followed in the final grade.

References

[1] Harris, J. A. *Schaum's Outline of Theory and Problems of Operating Systems*. McGraw-Hill, USA, 2002.

[2] Silberschatz, A., Galvin, P., Gagne, G. *Operating Systems Concepts Essentials*. John Wiley & Sons, Inc., USA, 2011.

[3] Task Manager Overview [Online]. Available: http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/taskman_whats_there_w.mspx [Accessed: Mar. 15, 2011].

[4] Making the Most of Activity Monitor [Online]. Available: http://mac.appstorm.net/how-to/os-x/making-the-most-of-activity-monitor/ [Accessed: Mar. 15, 2011].

[5] What's Running, your system information utility [Online]. Available: http://www.whatsrunning.net/ [Accessed: Mar. 6, 2011].

[6] Sysinternals Process Utilities [Online]. Available: http://technet.microsoft.com/en-us/sysinternals/bb795533 [Accessed: Mar. 6, 2011].