My Insanely Great Operating Systems Research Paper

Geordi LaForge

Abstract

Describe your paper in 100-200 words, give or take. The command-line `wc` utility is really useful here!

Contents

1 Outline for Option 1: In-Depth Study

The following subsections form the outline for an in-depth study paper — or, "learn about an operating system aspect from top to bottom," from concept and theory all the way to implementation and sample code. In the final paper, these subsections should actually promoted to full sections, and this artificial umbrella section should be removed.

Sample topics for this type of paper are listed below. Accompanying citations represent sample starting points for reading about the corresponding topic:

- A particular operating system's I/O subsystem (e.g., [Com05])
- A particular operating system's kernel, with emphasis on what is distinctive or unique compared to other kernels (e.g., [RS01])
- An interesting operating system-related application or enhancement (e.g., [Bes05])
- A complete study of a specialized operating system (e.g., [Wil04])

A word of caution: because operating systems tend to be very well-documented, this type of paper will usually have few references, and you might find yourself simply restating a lot of the content that you have read. *Always* state things in your own words; it will probably help if you adhere *strictly* to the outline below, so that you are forced to comprehend and restructure the material sufficiently.

Of course, you can get an excellent starting point for most core concepts from the textbook, so you will probably be citing that as well [SGG05].

1.1 Introduction

Summarize the operating system aspect or feature that you are studying. The introduction should be written with the assumption that the reader is technically competent but might not know any specifics at all on this particular operating system concept, service, or construct.

1.2 Previous Work

Provide background work or prior versions (or equivalents) of your chosen subject matter. For this type of paper, "previous work" includes but is not restricted to: equivalent constructs from prior versions of the operating system; equivalents from other operating systems entirely; and earlier prototypes, research, or theory that led to the current version of your chosen subject matter. Most of your citations and references will probably reside here. Refer to the first few paragraphs in Section 2 for pointers on creating effective bibliographies and citations with BibTeX and LaTeX.

1.3 Concept and Theory

This section should describe your subject matter at a conceptual level: what is it, what are its core ideas, what are its primary functions. Definitions and overall schematics or diagrams belong in this section. If

some theory is involved (e.g., algorithms, mathematics, abstractions), this would be the right place to discuss that as well.

1.4 User View

Describe a user's perspective of your chosen subject: what does the user see? What benefits or functions does it provide? How does the user interface look, if any? Are there any flaws or issues in what is presented to the user?

In many respects, this section functions like a condensed "instruction manual" for a non-developer user.

1.5 Developer View

In this section, shift to a developer's perspective: how does one programmatically access your chosen subject? What features are provided? How are the concepts or abstractions expressed as programming constructs: data structures, interfaces, etc.? How are error conditions and logging handled? What capabilities would applications or other programs gain by using your chosen subject at a programmatic level?

1.6 Implementation

The final section discusses the implementation of your chosen subject. If possible, acquire the source code for your chosen subject so that you can really study "where the rubber meets the road." What language (you may need to be really specific here; typically just saying "C" won't be enough) was used in the implementation? Are there any interesting or particularly elegant design choices? Effective optimizations or algorithms? How bound is the implementation to the hardware?

1.7 Conclusion

Summarize your overall in-depth study. List any advantages and disadvantages, at any level, that you discovered. Provide suggestions for addressing the weaknesses that you found. Point to possible future work or directions, whether "official" or based on your own understanding.

1.8 (as an Appendix) Sample Code

A quality in-depth study should include sample code which you've tested, experimented, and played with; include those in an appendix. If needed, provide screenshots or other artifacts of the programs, since I may not always be able to try them firsthand.

2 Outline for Option 2: Research Report

The following subsections form the outline for a research report — essentially "read up on something, then report on what you read." In the final

paper, these subsections should actually promoted to full sections, and this artificial umbrella section should be removed.

This type of paper is likely to be the heaviest user of LaTeX's excellent bibliography features. Citations and references are handled automatically by LaTeX through its companion program, BibTeX. All you have to do is provide a bibliography file that provides the reference information and internal keys (very much like variable names) that you use in your document.

BibTeX supports virtually all kinds of references, including books [SGG05, Com05, Wil04], parts of books [Bes05], articles [Loe05, RS01], and conference proceedings [SG05, HJLT05, ZCT+05], to name a few. If not already included in your LaTeX distribution, download and install the `url` package to support formatting of URLs; you can usually mention these in the *note* or *howpublished* fields of your BibTeX file.

Sample topics for this type of paper are listed below. Accompanying citations represent sample references about the corresponding topic:

- New features beyond the classic process, memory, storage, and I/O responsibilities of a traditional operating system (e.g., better file searches [SG05], enhanced security features [CCC+05])
- New work or enhancements on a known or established aspect of an operating system (e.g., new approaches to operating system processes and threads [Loe05, EKV+05], new power management techniques for mass storage [ZCT+05])
- New approaches to operating system design or construction (e.g., build an operating system with Haskell monads [HJLT05]!)

2.1 Introduction

Provide an introductory description of the operating system area that you researched in this paper. Discuss any goals, motivation, or examples of the subject; the key is to provide the reader with any information that is necessary to understand the project or work. This descriptive section should also allow the reader to understand the subsequent detail sections on the subject.

2.2 Background, Preliminary, and Related Work

Describe any history, work, or projects that serve as direct predecessors to the subject that you are summarizing or surveying. These items typically form the context or environment that spawned the subject of your paper. Previous work that has since been replaced or supplanted by the subject of your paper belongs in the section as well. Look at the sample research papers to see how they organized, presented, and discussed prior work.

The bibliographic notes in the text [SGG05] provide some pointers to seminal or key works; because they made it into the textbook they aren't necessarily "bleeding edge," but they likely provide the foundation for your chosen subject matter.

2.3 User View

Describe a user's perspective of your chosen subject: what does the user see? What benefits or functions does it provide? How does the user interface look, if any? Are there any flaws or issues in what is presented to the user?

Since your subject matter for this type of paper is "bleeding edge" work, much of this may be speculative or hypothetical; that's OK. What matters is that you are able to communicate how your chosen subject will affect its end-users when/if it reaches them in some final form.

2.4 Developer View and Implementation

In this section, shift to a developer's perspective: how does one programmatically access your chosen subject? What features are provided? How are the concepts or abstractions expressed as programming constructs: data structures, interfaces, etc.? How are error conditions and logging handled? What capabilities would applications or other programs gain by using your chosen subject at a programmatic level?

Again, since your subject matter for this type of paper is not necessarily "finished goods," this section may also be tentative or prototypical. As with the user view, the important part is that you are able to communicate how developers would interact with or use your chosen subject.

For the same reason, implementation details may be few and far between; still, a well-reported research area will have discussed this issue, and your findings in this area would belong in this section.

2.5 Current Status and Discussion

Provide the most up-to-date status report on the subject of your paper — what it can do, its level of maturity, any current applications, any related work, the latest findings or observations — complete with citations to the sources that provided this information. A key component of this section with be any references to the most recent publications, documents, or version of your paper's subject.

Another key component of this section is an evaluation of the subject: successes, failures, problems solved, and problems discovered. This component serves as an excellent transition to the final major section, which is...

2.6 Future Directions

Summarize what authors or developers have said about the future direction of your paper's subject. What's next? What new features are planned? What technical challenges or other barriers lie ahead?

2.7 Conclusion

Wrap up your survey with an "executive summary" of the paper's subject, its background, its current status, and its future directions.

2.8 (as an Appendix) Sample Code

The ability to include sample code will depend greatly on the maturity level of your chosen topic, so this section may or may not apply to the paper. However, if it is available (even if it is speculative or just a "projection"), this would be where it would go.

As with the in-depth study, provide screenshots or other artifacts of the programs (which may be mockups for this type of paper).

References

[Bes05] Steve Best. *User-Mode Linux*, chapter 11. Prentice Hall, 2005.

[CCC+05] Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, and Paul Barham. Vigilante: end-to-end containment of internet worms. *SIGOPS Oper. Syst. Rev.*, 39(5):133–147, 2005.

[Com05] Apple Computer, Inc. *I/O Kit Fundamentals*. Apple Computer, Inc., 2005. `http://developer.apple.com/documentation/DeviceDrivers/Conceptual/IOKitFundamentals`.

[EKV+05] Petros Efstathopoulos, Maxwell Krohn, Steve VanDeBogart, Cliff Frey, David Ziegler, Eddie Kohler, David Mazières, Frans Kaashoek, and Robert Morris. Labels and event processes in the asbestos operating system. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 17–30, New York, NY, USA, 2005. ACM Press.

[HJLT05] Thomas Hallgren, Mark P. Jones, Rebekah Leslie, and Andrew Tolmach. A principled approach to operating system construction in Haskell. In *ICFP '05: Proceedings of the tenth ACM SIGPLAN international conference on Functional programming*, pages 116–128, New York, NY, USA, 2005. ACM Press.

[Loe05] Keith Loepere. Stackable thread mechanisms. *SIGOPS Oper. Syst. Rev.*, 39(4):4–17, 2005.

[RS01] Mark Russinovich and David Solomon. Windows XP: Kernel improvements create a more robust, powerful, and scalable OS. *MSDN Magazine*, 16(12), December 2001.

[SG05] Craig A. N. Soules and Gregory R. Ganger. Connections: using context to enhance file search. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 119–132, New York, NY, USA, 2005. ACM Press.

[SGG05] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons, 7th edition, 2005.

[Wil04] Greg Wilson. *Exploring Palm OS: Memory, Databases, and Files*. PalmSource, Inc., 2004.

[ZCT+05] Qingbo Zhu, Zhifeng Chen, Lin Tan, Yuanyuan Zhou, Kimberly Keeton, and John Wilkes. Hibernator: helping disk arrays sleep through the winter. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 177–190, New York, NY, USA, 2005. ACM Press.