



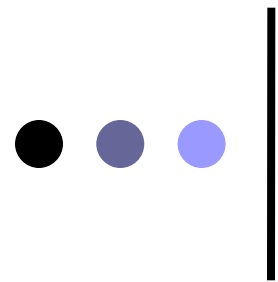
# How to Write a 6.033 Design Report

**Mya Poe<sup>1</sup> and Keith Winstein<sup>2</sup>**

<sup>1</sup> MIT Program in Writing and Humanistic Studies

<sup>2</sup> CSAIL

March 2006

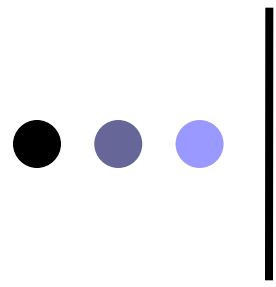


# Why are you here today?

1. Proposal → report
2. Show you how a computer designer “thinks through” a design problem.
3. Explain what we look for when grading your DP1.

# ● ● ● | Why are you here today?

1. Proposal → report
2. Show you how a computer designer “thinks through” a design problem.
3. Explain what we look for when grading your DP1.
  - Proposal is not the report!
  - “A” on proposal may not = “A” on report



# When thinking about this project at the meta-level . . .

- **Tech Audience:** Engineers implementing filesystem
- Purpose: Help the **Implementers**
- **Persuasive:** How does your design perform?  
How do you know?

- 
- √ Explain why you made decisions
  - √ Acknowledge design trade-offs
  - √ Use figures! (stand-alone visuals)
  - √ Write for readers who do not read chronologically.
  - √ Analysis is key! Be persuasive. Use data.



# Steps in the Writing Process

1. Read comments on your proposal
2. Re-read the assignment
3. Prioritize issues + get feedback (e.g., from friend)
4. Write
5. Double-check assignment
6. Clarify and refine report -- peer review!
7. Proofread

Step  
#1

## Read comments on your proposal

- What information was missing or unclear?
- What was good?
- Can you build off existing design or do you need to “start from the ground up”?

## **You wrote:**

“JoeFS keeps a list of available blocks, with their length, on the disk. When creating a new file, JoeFS finds the smallest contiguous slice that is bigger than the length of the file.”

## **You wrote:**

“JoeFS keeps a list of available blocks, with their length, on the disk. When creating a new file, JoeFS finds the smallest contiguous slice that is bigger than the length of the file.”

## **TA responded:**

**How do you maintain this list correctly in memory? How do you know the file size in advance? (open() does not tell you the file size.)**





Better:

“JoeFS divides the 120 GB filesystem up into 1 MB chunks, and uses a bitmap in RAM to record whether each chunk is occupied or not. JoeFS always waits until 1 MB has been written before storing the megabyte into a chunk. Therefore, JoeFS never has to seek more often than once per megabyte.”



Best:

- What about small files?
- What is the step-by-step process taken on `open()`, `read()`, `write()`, `close()`, and `unlink()`?
- Use diagrams to show how data structures evolve.
- What actual throughput will your filesystem achieve on the three workloads?

**Step  
#2**

# Re-read assignment to find information missing from proposal

Proposal did not address all aspects of the assignment:

- What's missing?
- What about format? Document specs
- FAQ: Check daily

**Compile list of issues:**

**Comments + assignment + FAQ**



# Pitfalls:

1. Not analyzing the performance of FS on the 3 workloads:
  - If you need to make assumptions, make them and justify them.
  - If you need to do simulations, do them.
  - If you need data, use your own hard drive.
  - But you need to tell us the throughput you will achieve (at least in the average case) on those workloads!
2. Not describing precisely what your FS does for `open()`, `read()`, `write()`, `close()` and `unlink()`. Remember that your audience = people actually implementing the filesystem.
3. Not including good diagrams of data structures and the processes that maintain them.
4. Vague language like, “Tries to keep files together.” Design a system that does this – don’t just hope it works.

## Step #3

# Identify priorities for your design

1. Is it simple to explain? Is it easy to analyze? How does it perform on the three workloads?
1. Could two programmers implement it from your design report and achieve compatible implementations? Aspire to this. What do you actually do on `open()`, `read()`, `write()`, `close()`, and `unlink()`?
1. Are there large gaps in your explanation? You cannot say something like, “When writing a new file, the system performs the minimal defragmentation necessary to fit it contiguously,” without analyzing the performance implications of this choice.
1. Your design might be too complicated to analyze (the above probably is)! If so, simplify! As a last resort, run simulations.

## Step #4a

# The Design Introduction overviews your design goals and approach

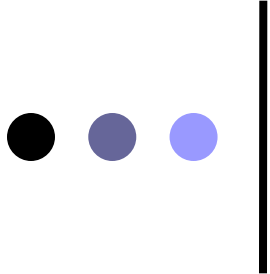
- State your design approach
- Trade-offs
- Rationale
- Analysis Results

---

### **Example Template**

#### **1.0 Design Overview**

The goal of this design is to provide . . . We accomplish this goal by . . . .



**Introduction:** Summarize the salient aspects of your design, the trade-offs you made, a brief rationale for the design you have chosen, and the results from your analysis

---

## 1.0 Design Overview

SuperUnixFS is a simple, high-performance filesystem modeled after the Unix file system with minor changes. By writing files only in large chunks and storing metadata in RAM, SuperUnixFS achieves 95% efficiency on typical workloads, compared with Unix's 40% efficiency. Additionally, SuperUnixFS is immune to fragmentation, so its performance will not degrade with time. These optimizations come at a cost, however: SuperUnixFS wastes disk space when storing small files.

## Step #4b

# The Design Description details your design approach

- Organize by topic:

### Draft Subheads

- 2.1 Data Structures used (in RAM and on disk) with diagrams.
- 2.2 What happens on open(), read(), write(), close(), unlink().
- 2.3 Several worked-thru examples with diagrams.
- 2.4 How does it perform on the sample workloads? Use numbers! Use data. Do a non-BS analysis.

- Use subsections to show hierarchy of ideas
- Tell readers what & why you made choices
- Weave in discussion of design trade-offs



## Step #4b

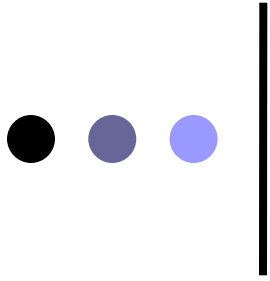
# The Design Description details your design approach

- Organize by topic:

### Final Subheads

- 2.1 How Files are Represented
- 2.2 Implementation of System Calls
- 2.3 Common Workflows
- 2.4 Performance Analysis

- Use subsections to show hierarchy of ideas
- Tell readers what & why you made choices
- Weave in discussion of design trade-offs



## Example

### 2.3 Web Server Process Architecture

The web server uses a SPED architecture for its simplicity and performance. Because the system exclusively uses RAM for data-storage, there was no need to worry about kernel support for asynchronous disk I/O or other disk-related drawbacks that SPED may have. On the other hand, a SPED design makes it easy to implement the web server functionality needed for this system.

Courtesy of Vincent Yeung



## Develop description from general to specific

### Example

#### 2.3 Web Server Process Architecture

The web server uses a SPED architecture for its simplicity and performance. Because the system exclusively uses RAM for data-storage, there was no need to worry about kernel support for asynchronous disk I/O or other disk-related drawbacks that SPED may have. On the other hand, a SPED design makes it easy to implement the web server functionality needed for this system.

Topic sentence  
conveys purpose of ¶



What &  
why of  
design  
decisions  
explained

Courtesy of Vincent Yeung

## Use figures & pseudo-code to illustrate concepts

The `delete_protection_domain` works analogously to `grow_protection_domain`. When a process is deleted from the mote, all the data that followed that process in memory is “compacted”, or shifted up, creating one large block of allocated memory instead of two smaller separated blocks. For example, if a process with seven pages of allocated memory is destroyed, the OS shifts all the other data in memory up by seven pages, as shown in Figure 8. The pseudocode below details the operation of the `delete_protection_domain` procedure.

```
boolean destroy_protection_domain([in] int PID)
{
    int removed_pages ← bound_table[PID] + 1;

    // update the base_table values and the EOMpointer
    EOMpointer ← EOMpointer - removed_pages;

    for (p = 0; p < max_processes; p++)
    {
        if (base_table[p] > base_table[PID])
            base_table[p] = base_table[p] - removed_pages;
    }
}
```

*Figure 8. Pseudocode for the `destroy_protection_domain` method. Note that there is no need to explicitly reset the values stored for the base and bound of the destroyed PID. When a new process is given that PID, the old values will be overwritten anyways.*

Courtesy of Rohit Rao.

## Use figures & pseudo-code to illustrate concepts

The `delete_protection_domain` works analogously to `grow_protection_domain`. When a process is deleted from the mote, all the data that followed that process in memory is “compacted”, or shifted up, creating one large block of allocated memory instead of two smaller separated blocks. For example, if a process with seven pages of allocated memory is destroyed, the OS shifts all the other data in memory up by seven pages, as shown in Figure 8. The pseudocode below details the operation of the `delete_protection_domain` procedure.

```
boolean destroy_protection_domain([in] int PID)
{
    int removed_pages <- bound_table[PID] + 1;

    // update the base_table values and the EOMpointer
    EOMpointer <- EOMpointer - removed_pages;

    for (p = 0; p < max_processes; p++)
    {
        if (base_table[p] > base_table[PID])
            base_table[p] = base_table[p] - removed_pages;
    }
}
```

*Figure 8. Pseudocode for the `destroy_protection_domain` method. Note that there is no need to explicitly reset the values stored for the base and bound of the destroyed PID. When a new process is given that PID, the old values will be overwritten anyways.*

## Step #4c

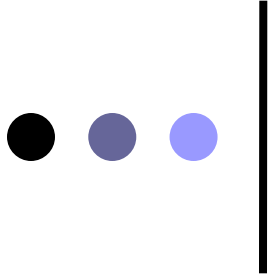
# Write conclusion

- Summarize design problems you solved,
- Identify problems in your design, &
- Identify further actions

## Example Template

### 5.0 Conclusion

This design uses [x] to . . . This design does not cope well with [x] because . . . [explain why you did not address this issue]



**Conclusion:** Provide a short conclusion that provides recommendations for further actions and a list of issues that must be resolved before the design can be implemented.

---

## 5.0 Conclusion

SuperUnixFS, an enhanced version of the Unix file system, provides acceptable performance with low complexity on many common workloads. However, because SuperUnixFS inflates all files to at least one megabyte in size, this filesystem is not appropriate for workloads with many small files. Future development should focus on improving SuperUnixFS in order to deal with these cases.

**Step  
#4d**

Write the front and end matter



- **Title Page**

Title

Your name

ID#

Recitation instructor

Section time

Date

- **Acknowledgements**

= anyone who helped you  
with your design

- **References**

IEEE style



## **Acknowledgements**

Thank you to Professor Kaashoek and Chris Lesniewski-Laas for their suggestions on achieving fault isolation.

## **References**

[1] F. Cavalieri, T. Ruscio, R. Tinoco, S. Benedict, C. Davis, and P. K. Vogt, "Isolation of three new avian sarcoma viruses: ASV9, ASV17, and ASV 25," *Virology*, vol. 143, pp.680-683, 1985.

## Step #5

A little extra time dedicated for review will improve your grade

- Give your report to a peer for review
- Double-check the design specs
- Consider from the audience perspective.  
“I’ve gotten a disk formatted with your filesystem – how do I write the Linux driver in order to read and write it?”

## Step #6

# Proofreading Checklist

- Did you # the pages?
- Is your name on every page?
- All figures/tables labeled & referenced in the text?
- All sources cited?
- Did you avoid:
  - naked “this”
  - “the reason is because . . .”
  - “the fact that . . .”
  - over-use of “I”
  - “due to” is an adjective. Try “because”
  - passive voice
- Did you proofread a printed copy?



# Report Format

- 11 or 12 point font
- No more than 2,500 words
- Submit 2 copies
- Single-side printed
- Color, not required



# Writing Help

- Model DP1 papers on 6.033 website
- Readings in your course packet
- Writing Center <http://web.mit.edu/writing>
- *Mayfield Handbook of Technical and Scientific Writing*

- Writing Tutors available:

Contact your recitation grader for an appointment



# How do we grade DP1?

## **Technical staff:**

1. Is the design described unambiguously?
2. Is the design's performance well-analyzed?
3. Are your design decisions and analysis assumptions well justified?



# How do we grade DP1?

## **Technical staff:**

1. Is the design described unambiguously?
2. Is the design's performance well-analyzed?
3. Are your design decisions and analysis assumptions well justified?

## **Technical and Writing Staff:**

1. Is the report well-organized within and across sections?
2. Is the report professionally presented?
3. Are text and figures integrated?
4. Is the writing crafted for readability? Proofread?