6.033 2013 Design Project 1: A versioning file system

I. Due Dates and Deliverables

There are three deliverables for Design Project 1:

1. A design memo, due on Feb 15, 2013 (before tutorial).
2. A design proposal not exceeding 800 words, due on March 1, 2013 at 5pm.
3. A design report not exceeding 2,500 words, due on March 22, 2013 at 5pm.

All deliverables should be submitted via the online submission site.

As with real-life system designs, 6.033 design projects are under-specified, and it is your job to complete the specification in a sensible way given the overall requirements of the project. As with designs in practice, the specifications often need some adjustment as the design is fleshed out. We recommend that you start early so that you can evolve your design over time. A good design is likely to take more than just a few days to put together.

II. The Problem

Your goal is to design a versioning file system. Many popular operating systems have a versioning file system (sometimes called continuous snapshotting file systems), which stores all versions of each file over time. These designs are complex because they handle a wide range of failures. Your task is simpler: design a versioning file system that assumes no failures. That is, you can assume that the file system will always terminate cleanly, even in the case of unexpected failures. For instance, if a power failure shut down the computer, assume that the file system will get a chance to write its state to stable storage cleanly. This allows you to design a much simpler file system than commercial ones. (After the break, we will study designs that can handle failures.)

When a program writes a new version of a file, your design should keep track of the old and new versions of the file. That is, a user should be able to see the old version in the file system namespace, and read it (but not write it). Your main challenge will be to do this in a storage-efficient manner: if, for example, a program changes one block of the file, then your design should use disk space proportional to 1 copy of the file plus one additional disk block.

Your design should support the standard Unix file system operations (read, write, open/create, rename, link, unlink, symlink, mkdir, chdir and stat). Your design can

also be slower in performance than a non-versioning file system, but it should not be unusably slow. Section 2.5 (Case study: UNIX file system layering and naming) of the text book describes the basic design of non-versioning Unix file system. Your design should allow the user to specify files and directories that shouldn't be versioned. Finally, your design should be able to support garbage collection of older versions of a file.

Your design paper should clearly describe your design and the *on-disk* and *in-memory* data structures for snapshotting, and how they are used to support the required operations.

III. Requirements

Your system must support the following use cases:

- Writing a few blocks of a large file
- Appending to a log file
- Creating a new file in a large directory
- Finding some string across all versions of a file
- Finding some string across all versions in a file system
- Allowing users to exclude large files that change often from versioning (e.g., virtual machine image or a database file)
- Allowing users to exclude specific directories from versioning (e.g, "/tmp").

In supporting these use cases here are a few general issues to think about:

- How do you represent files on disk? If an application changes one block of a large file, the new version should require some small constant number of additional disk blocks (e.g., 1).
- What is the policy for when to create a new version? When an application closes a file? When an application execute a write system call?
- Should directories be treated differently than ordinary files?
- How do users specify which files/directories are excluded from versioning?
- How do you name versions of a file?
- How do users specify searches across many versions?

This list of issues is incomplete, but hopefully helps you get going.

You should analyze your solution under three workloads both in terms of number of disk blocks read and amount of space used. The three workloads are: 1) repeatedly writing to a small file; and 2) repeatedly writing a block of a large file; and 3) searching through all versions of a small file.

Challenge: save space by extending your design to do *deduplication* across files: if a block appears twice in different files store it once. Addressing this challenge is not necessary to get an A, but may be fun.

IV. Design Proposal

The design proposal should be a concise summary (800 words) of your overall system design.

The core of the proposal should be the design description. This section must include at least one graphic, correctly formatted with a caption and brief description. For example, it may be a good idea to illustrate the design of your data structures used to store versions.

You do not have to present a detailed rationale or analysis in your proposal. However, if any of your design decisions are unusual (particularly creative, experimental, or risky) or if you deviate from the requirements, you should explain and justify those decisions in your proposal.

You will receive feedback on your proposal from your TA in time to adjust your final report. You will also receive a writing program grade for your proposal, as well as feedback from your writing instructor that will help you improve the writing of the final report. Your writing instructor will evaluate the proposal according to the CI guidelines.

Some writing considerations for the proposal (and report):

- Does the proposal/report summarize the proposed system as a whole?
- Does the proposal/report make an argument in favor of the system (as opposed to a different design)?
- Is content organized in accordance with the guidelines?
- Do hierarchy and structure match guidelines and clearly signal the role of information?
- Do figures clarify or illustrate an essential concept presented in the text? Are figure properly formatted, labeled, and referenced?
- Does the document meet basic standards of professional grammar, syntax, spelling, etc?

Here are a few tips:

- Use ideas and terms from the textbook and papers when appropriate; this will save you space (you can refer the reader to the relevant section of the textbook) and will save the reader some effort.
- Before you explain the solution to any given problem, say what the problem is.
- Before presenting the details of any given design component, ensure that the purpose and requirements of that component are well described.
- It's often valuable to illustrate an idea using an example, but an example is no substitute for a full explanation of the idea.
- You may want to separate the explanation of a component's data structures from its algorithms to access or use those data structures.
- Explain all figures, tables, and pseudo-code; explain what is being presented, and what conclusions the reader should draw.

V. Design Report

Your report should explain your design. It should discuss the major design decisions and tradeoffs you made, and justify your choices. It should discuss any limitations of which you are aware. You should assume that your report is being read by someone who has read this assignment, but has not thought carefully about this particular design problem. Give enough detail that your project can be

turned over successfully to an implementation team. Your report should convince the reader that your design satisfies the requirements in Section III.

V.A. Report organization

Use this organization for your report:

- Title page: Give your report a title that reflects the subject and scope of your project. Include your name, email address, recitation instructor, section time(s), and the date on the title page.
- No table of contents is needed.
- Introduction: Summarize what your design is intended to achieve, outline the design, explain the major trade-offs and design decisions you have made, and justify those trade-offs and decisions.
- Design: Explain your design. Identify your design's main components, state, and algorithms. You should sub-divide the design, with corresponding subsections in the text, so that the reader can focus on and understand one piece at a time. Explain why your design makes sense as well as explaining how it works. Use diagrams, pseudo-code, and worked examples as appropriate.
- Analysis: Explain how you expect your design to behave in different use cases. What use cases might pose problems for throughput, latency, or even correctness? What do you expect to be the scalability limits of your design?
- Conclusion: Briefly summarize your design and provide recommendations for further actions and a list of any problems that must be resolved before the design can be implemented.
- Acknowledgments and references: Give credit to individuals whom you consulted in developing your design. Provide a list of references at the end using the IEEE citation-sequence system ("IEEE style") described in the Mayfield Handbook.
- Word count. Please indicate the word count of your report at the end of the document. Captions of figures should be included in the total word count.
- Footnotes. Please do not use footnotes in your report.

The writing suggestions for the proposal also apply to the report.

V.B. How we evaluate your work

Your recitation and writing instructors will assign your report a grade that reflects both the design itself and how well your report presents the design. The most important aspect of your design is that we can understand how it works and that you have clearly addressed the requirements and provided the elements listed in Sections III and V. Complicated designs that we cannot understand will not be graded favorably.

Some overall content considerations:

- Design supports use cases and describes how they work.
- Clean yet sufficient design, minimal mechanism.
- Reasonable implementation detail.
- Convincing space and performance evaluation.

The grading rubric for the final report is as follows:

Overall design • When implementation details are presented are they justified with reference to the overall purpose of the system? • Is there any consideration of possible design alternatives and why the resulting ones were chosen?	30
The degree to which the design addresses the requirements and use cases • Does the paper discuss all the stipulated use cases and demonstrate the design meets the requirements?	20
Analysis of space and time requirements • Is the analysis quantitative and justified in terms of reasonable performance metrics for the underlying components (disk, network, processing, ...)	20
User experience • Is the system behavior described in terms of what users would experience?	15
Quality of the figures that illustrate the design	5
Overall presentation	10

The items in the grading rubric are not independent: a design that we cannot understand will likely result in a low score for several items.

85 and above is an A grade. Between 60 and 85 is a B grade. You will have to hand in a design project to pass 6.033.

VIII. Collaboration

This project is an individual effort. You are welcome to discuss the problem and ideas for solutions with your friends, but if you include any of their ideas in your

solution you should explicitly give them credit. You must be the sole author of your report.

IX. Clarifications

- Your design does not need to deal with failures (e.g., power outages, disk failures, etc.), with security concerns (e.g., user changing ownership of files and directories), or with distributed aspects.
- You can find previous examples of DP1 reports under the "Excellent Writing Examples" link on the left-hand side of the course home page.
- *Late submission grading policy*: Your DP1 report is due before 5pm on March 22, 2013. If you submit your DP1 report late, we will penalize you one letter grade per 48 hours, starting from 5pm on the submission day. For example, if you submit the report anywhere from 1 minute to 48 hours late, and your report would have otherwise received a grade of "A", you will receive a "B"; if you submitted 49 hours late, you would receive a "C".