Project Guidelines and Schedule

Intro

You have the choice of either writing a survey paper or doing an implementation project. Survey papers are written individually whereas implementation projects should be done with a partner if possible.

You can use the Discussion Board on our Blackboard site at learn.vt.edu if you have a project idea and are looking for a partner to share it with.

To ensure progress towards the completion of your project or paper, your team will meet with the instructor during certain milestones, listed below. Those meetings are mandatory for full credit on the project.

Timeline

Sep 26	Initial proposal (max 2 pages) due. For survey papers, submit a description that describes your initial research on the topic, including a list of literature you will study. For a survey paper, I insist that you use a word processing system that separates the structure of your text from the formatting, such as LaTeX or Microsoft Word Templates. For projects, describe the goal of your project (what are you going to show?), the relevance (why is this important?), the major components and the technology infrastructure you plan on using. Include a plan of work. Outline what you will have accomplished by milestones 1 and 2 (see below.) You should have started initial investigations by that time.
Sep 27-Oct 7	Work with instructor to revise and get approval for topic.
Oct 24	*Survey:* Outline of paper due. In addition, create and submit written summaries of the sources you are researching. *Project:* Progress report 1. (1 page)
Oct 25-28	Meet formally with instructor to discuss progress.
Nov 11	*Survey:* First complete draft of paper due.

	Project: Progress report 2. (1 page)
Nov 14-17	Second formal meeting with instructor to discuss progress.
Dec 8	Final survey paper due.
Dec 13	Final project report due.
Dec 8-15	Final presentations.

Survey Paper Suggestions

Instead of merely summarizing a number of papers, a *survey paper* should explore a particular research area or controversy. It should include a historical perspective that explains the relevance of the chosen research topic. It should outline the design space, compare and contrast multiple different approaches to a problem and discuss their trade-offs.

Ideally, you would pick a survey paper in an area that is related to your research interests.

The following are some possible topics for a survey paper:

- Threads vs. Events. An old, unresolved controversy is which of these two models is better suited for designing concurrent systems. Since Lauer & Needham's paper in 1979, which was supposed to settle the issue, different researchers and systems designers have provided arguments and implementations favoring on or the other. Describe the controversy, and describe how changing assumptions and needs influenced the design of concurrent systems.
- Hardware vs. Software Protection. Traditional OS implement protection in hardware using distinct address spaces. During the last 15 years, several techniques have emerged that promise to implement at least some aspects of protection in software, such as software-fault isolation, type-safe languages, proof-carrying code. Compare and contrast these approaches.
- Virtual Machines. Virtual Machines, originally developed in the 70s, have recently seen a comeback. They are used for both desktop PCs and large servers. Survey the different techniques, paying particular attention to the degree to which hardware is virtualized and the trade-offs involved.

Additional topics that have recently received interest in the OS community include

- User-level vs. kernel-level threading and hybrids
- Real-time scheduling techniques
- Ubiquitous/Pervasive computing
- Multi-tasking support for language-based virtual machines
- Applications of static program analysis to OS
- File Systems (metadata handling and recovery)
- Power and energy management
- New approaches to system administration and configuration
- Overlay networks
- OS support for homogeneous clusters
- Scalable servers (web servers, web caches)
- Structured peer-2-peer distributed systems

Project Ideas

A *project* can take multiple forms and you have wide latitude in designing your specific project.

Your project can provide systems support for or solve a systems-related problem in your research area, but it can also involve an application that uses systems techniques.

You are encouraged to come up with your own ideas for a project that is related to your research interests and discuss them with the instructor. This is your chance to take the time and get the support and even credit for a system project you've always wanted to do!

Projects students have done in past years include

- A scalable, distributed implementation of Tuple Spaces.
- A network audio server that distributes audio streams to multiple destinations.
- Real-time support for Linux sound drivers.
- A decentralized P2P system.
- A distributed agent system.

Additional ideas for projects include

- Policies for proportional-share scheduling. Identify what policies are applicable to such schedulers as VTRR.

- Analyzing and visualizing application cache-behavior. Create a tool or adapt existing tools that analyze and visualize a program's cache-behavior.
- Your own device driver for a device you wish to have support for.
- A library that solves a domain-specific problem: for instance, a library that provides transparent access to files using a high-speed inter-connect.
- A distributed system. Learn how to use any of the common technologies (client/server, CORBA, RMI, JXTA, JavaSpaces/JINI, .Net, Web Services) to prototype a small distributed system, such as a file/storage system, a chat system, a distributed game. A very interesting project is also to take an existing application and create a distribute version of it.
- Provide systems-support for location-aware services, such as SeeVT.
- A parallel system. Parallelize a specific program or algorithm.
- User-level file systems: implement application-specific caching policies.
- Soft real-time systems: Provide support for TUF (time-utility functions) scheduling in MVM or Jikes.
- Checking systems software. Learn how to use and extend checking software to check systems software for bugs.
- Binary rewriting: Use binary rewriting tools to rewrite systems software to perform runtime verification and checking. An example of this might be software that implements Eraser's algorithm.

For all projects, it is important that the focus lies on the systems aspects (that is, those aspects that are common to or can be shared by multiple applications) rather than the logic or GUI of a specific application.