

# CS162 – Section # 2, 02/04/2003

Rodrigo Fonseca

---

## Announcements

Get familiar with the environment.

CVS Help session:

Feb. 5th (tomorrow), 6 pm  
306 Soda

Upcoming deadlines:

Proj 1 Initial Design due Wed, 2/19 (2 weeks from  
tomorrow)

---

## Infra-structural issues:

### SSH key

- The ssh generation should be run automatically on your first login.
- You need to have a identity.pub file in your .ssh directory.
- Do not keep regenerating the key, as it is the authentication means for CVS

### CVS

- Groups should use CVS for sharing and synchronizing code during projects
- Great way to manage the project (and also serves to back up everything you do!)

## Primer

CVS is used to keep track of collections of files in a shared directory called "The Repository". Each collection of files can be given a "module" name, which is used to "checkout" that collection.

After checkout, files can be modified (using your favorite editor), "committed" back into the Repository and compared against earlier revisions. Collections of files can be "tagged" with a symbolic name for later retrieval.

You can add new files, remove files you no longer want, ask for information about sets of files in three different ways, produce patch "diffs" from a base revision and merge the committed changes of other developers into your working files.

Help session TOMORROW!

---

## Key points of the week:

Processes, Threads and Address Spaces  
Concurrency and Protection  
Thread life  
Context Switching

Program vs Process - an analogy: a program is a script, a process the play

What is the advantage of multiple threads per address space?

When would you not want this?

---

## Thread Lifecycle

States:

NEW  
READY  
RUNNING  
WAITING  
TERMINATED

There are *internal* and *external* events that cause a thread to relinquish the CPU.

External events characterize preemption.

>> Draw the diagram. Which transitions from RUNNING are external and internal?

---

Dispatcher Loop

```
Loop {  
    run thread  
    *choose thread to run  
    switch  
}
```

The dispatcher can make scheduling decisions at different transitions in the diagram:

1. A thread switches from RUNNING to WAITING state (e.g., I/O, wait(), yield())
2. A thread switches from RUNNING to READY (e.g., interrupt)
3. A thread switches from WAITING (or NEW) to READY, (completion of I/O, arrival of new process)
4. A thread terminates

Which are non-preemptive scheduling?

Which are preemptive schedule?

In Non-preemptive scheduling, scheduling decisions are made only in situations 1 and 4, i.e., a new process is scheduled only when a CPU burst of the currently running process has finished.

---

To get a head start at Nachos AND understand context switching, take a look at the first homework at

<http://inst.eecs.berkeley.edu/~cs162/hw/hw1.html>

(This is neither required nor graded, but can be very instructive)

---