Reminders
- Form groups of 3 by tomorrow
  - Email group usernames to me
- Start project 2!
  - Read the assignment carefully
  - Read it again
- Today:
  - Project 2 intro
  - CVS

Project 2
- You have to:
  - Implement a user thread library
    - Part a
      - Implement synchronization primitives
      - Solve a synchronization problem
      - Implement a multithreaded web server
    - Part b
      - Analyze and experiment with your design
      - Write a report
- Part a due: October 26
- Part b due: November 4

Simplethreads
- We give you:
  - Skeleton functions for thread interface
  - Machine-specific code
    - Support for creating new stacks
    - Support for saving regs/switching stacks
  - A generic queue
    - When do you need one?
  - Very simple test programs
    - You should write more
  - Singlethreaded web server

Simplethreads Code Structure
- You write this

Thread Operations
- Which ones do we need?
  - void pthread_init()
  - Initialize the whole system
  - thread_t pthread_create(func start_func, void *arg)
  - Create a new thread and make it runnable
  - void pthread_yield()
  - Give up the CPU
  - void pthread_exit(void *ret)
  - Exit current thread
  - What about the TCB?
    - struct _thread {
      - thread_ctx_t *saved_ctx;
      - ...
    }
  - Others?
Sample multithreaded program

```c
int main(int argc, char **argv) {
    int i;
    sthread_init();
    for(i=0; i<4; i++)
        if(sthread_create(thread_start, (void*)i) == NULL) {
            printf("stthread_create failed!");
            exit(1);
        }
    sthread_yield();
    printf("Back in main\n");
    return 0;
}
```

Managing Contexts (given)

- `stthread_new_ctx(func_to_run)`
  - gives a new thread context that can be switched to

- `stthread_free_ctx(some_old_ctx)`
  - Deletes the supplied context

- `stthread_switch(old_ctx, new_ctx)`
  - Puts current context into old_ctx
  - Takes new_ctx and makes it current

How sthread_switch works

1. **Push old context**
   - `p pusha`
   - `movl %esp, (%eax)`
   - `movl %edx, %esp`
   - `popa`
   - `ret`

2. **Save old stack pointer**
   - `p pusha`
   - `movl %esp, (%eax)`
   - `movl %edx, %esp`
   - `popa`
   - `ret`

3. **Change stack pointers**
   - `p pusha`
   - `movl %esp, (%eax)`
   - `movl %edx, %esp`
   - `popa`
   - `ret`
### Pop off new context

- `thread_switch:
push %esp
movl %esp,(%eax)
movl %edx,%esp
popa
ret`

### Adjusting the PC

- `ret` pops off the new return address!

### Things to remember

- Your thread library is non-preemptive
- You can compare your implementation against pthreads (which is preemptive).
  - `./configure --with-pthreads`

### Things to think about

- Who will call `thread_switch`?
- Where does `thread_switch` return?
- How do we delete a thread?
  - Can a thread free its stack itself?
- Starting up a thread
  - `thread_new_ctx` doesn't pass parameters to the function it runs
  - How do you pass parameters to a function with no arguments?

### Done; return

- `thread_switch:
pusha
movl %esp,(%eax)
movl %edx,%esp
popa
ret`

- What got switched?
  - SP
  - PC (how?)
  - Other registers

### What is CVS

- Version control system for source files
- Multiple users can work on the same file simultaneously
Why use CVS

- The other way:
  - Keep every version of code, all with different names:
    - Project2prod
    - Project2_working
    - Project2_Feb_2_alex
    - Project2_old
  - Sends emails back and forth with new changes
  - Merge different versions by hand

- The CVS way:
  - One version, saved in the CVS repository
  - Multiple people can work on the same file concurrently
  - CVS merges the edited versions automatically as you put them back in the repository

Setting up CVS

- Set up CVS root
  - setenv CVSRoot /cse451/groups/cvs
  - (bash) export CVSRoot=/cse451/groups/cvs

- Initialize a repository (only one person per group)
  - cd /cse451/groups
  - mkdir cvs
  - cvs init

Setting up CVS (2)

- Add the simplethreads distribution
  - tar xvf simplethreads-1.10.tar.gz
  - cd simplethreads-1.10
  - cvs import -m "Initial code"
  - simplethreads SIMPLTHREADS
  - SIMPLETHREADS_1_10
  - cd ..
  - rm -fr simplethreads-1.10

CVS Commands

- Check out a project to your home directory:
  - cd ~
  - cvs checkout simplethreads

- Merge in new changes from repository (update):
  - cvs update [files...]

- Save your edited files into the repository:
  - cvs commit -m "fixed annoying bugs" [files...]

CVS Commands 2

- Add a new file to the repository
  - cvs add [files...]

- Check status of a file
  - cvs status file.c

- Check differences between your file and one in the repository:
  - cvs diff file.c
  - cvs diff -r 1.1 file.c (specifies version)

- View log of changes for a file
  - cvs log file.c

- More info:
  - http://www.cvshome.org
  - cvs -help-commands

CVS Remote Access

- Access CVS from another machine:
  - setenv CVSRoot
  - coredump.cs.washington.edu/cse451/cse451a/cvs
  - setenv CVS_RSH ssh

  (for CVS to know how to access repository)
  (add to ~/.login (csh) or ~/.profile (bash))