Reminders

- Start project 2!
  - It’s long
  - Read the assignment carefully
  - Read it again

- Project 2 will be done in groups of 3
  - E-mail groups to me

- Part 1 due in 2 weeks (10/27)
- Part 2 due in 3.5 weeks (11/7)
Project 2

- You have to:
  - Implement a user thread library
  - Implement synchronization primitives
  - Solve a synchronization problem
  - Add Preemption
  - Implement a multithreaded web server

Part 1

- Get some results and write a (small) report

Part 2
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
Structure

- test/*.c
- Web server (web/sioux.c)
- Other apps
- include/sthread.h

You write this
- lib/sthread_user.c

- lib/sthread_queue.h
- lib/sthread_ctx.h
- lib/sthread_preempt.h

lib/sthread_user.h
lib/sthread_queue.c
lib/sthread_ctx.c
lib/sthread_preempt.c
lib/sthread_switch.S

sthread_switch_i386.h
sthread_switch_powerpc.h
Thread operations

- void sthread_init()
  - Initialize the whole system
- sthread_t sthread_create
  - Create a new thread and make it runnable
- void sthread_yield()
  - Give up the CPU
- void sthread_exit(void *ret)
  - Exit current thread

- What about the TCB?
  struct _thread {
    sthread_ctx_t *saved_ctx;
    ......
  }

- Others?
Sample threaded program

```c
int main(int argc, char **argv) {
    int i;

    sthread_init();
    if (sthread_create(thread_start, (void*)i) == NULL) {
        printf("sthread_create failed\n");
        exit(1);
    }

    sthread_yield(); //yield main thread to our new thread
    printf("back in main\n");
    return 0;
}

void *thread_start(void *arg) {
    printf("In thread_start, arg = %d\n", (int)arg);
    return 0;
}
```
Managing contexts

- Thread context = thread stack + stack pointer
- `sthread_new_ctx(func_to_run)`
  - gives a new thread context that can be switched to
- `sthread_free_ctx(some_old_ctx)`
  - Deletes the supplied context
- `sthread_switch(oldctx, newctx)`
  - Puts current context into oldctx
  - Takes newctx and makes it current
Things to think about

- Who will call sthread_switch?
- Where does sthread_switch return?
- How do we delete a thread?
  - Can a thread free its stack itself?
- Starting up a thread
  - sthread_new_ctx() takes a function foo
  - sthread_new_ctx doesn’t pass parameters to foo
  - But in sthread_create, you give a function and an arg!
- **Bottom line:** how do you pass arguments to a function with no arguments?
Programming in groups

How to work on same files?

One way:

- Keep every version of code, all with different names:
  - Project2good
  - Project2_10_13_04
  - Project2working
- Send emails back and forth with new changes
- Merge different versions by hand
CVS

- 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
  - Maintains all old versions of files, so you can go back
Setting up CVS

- Set up CVS root environment var
  - Tells CVS where to find your repository
    - `setenv CVSROOT /cse451/groupleader/cvs`
    - *(bash)* `export CVSROOT=/cse451/groupleader/cvs`

- Initialize a repository (only one person per group)
  - Create a dir in your group’s dir to hold repository (master copy of code)
    - `cd /cse451/groupleader`
    - `mkdir cvs`
  - Initialize repository
    - `cvs init`
  - You now have an empty repository
Setting up CVS (2)

- Add/Import the simplethreads distribution to your repository
  - `tar xvfz simplethreads-1.20.tar.gz`
  - `cd simplethreads-1.20`
  - `cvs import -m "initial code" simplethreads SIMPLETHREADS SIMPLETHREADS_1_20`
  - `cd ..`
  - `rm -fr simplethreads-1.20`
CVS commands

- Check out a project (sandbox, or local copy) to your home directory to work on:
  - CVS creates a dir simplethreads/ and puts copy of all source files in repository into that dir
  - Also adds CVS dir where it stores data about what you check out
    - cd <wherever>
    - cvs checkout simplethreads
    - cd simplethreads

- Do this once

- Merge in new changes from repository (update):
  - cvs update [files...]
CVS commands (2)

- Save your edited files into the repository so others can use them:
  - `cvs commit -m "fixed annoying bugs" [files...]`
- Add a new file (source files .c or .h) 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](http://www.cvshome.org) or `man cvs`
CVS Miscellany

- Use emacs for cvs commit log editing (default is vi):
  - setenv VISUAL emacs

- Access CVS from another machine besides forkbomb?:
  - setenv CVSROOT forkbomb.cs.washington.edu:/cse451/groupleader/cvs
  - setenv CVS_RSH ssh
  (for CVS to know how to access repository – use ssh)