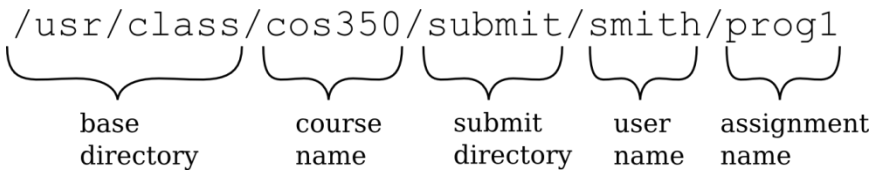
COS 350: Program #3: submit program

Objectives: Working with directories and file stat information.

You will write your own version of the unix lab submit program. The basic task of the submit program is to copy a student's files into a special directory to hold the submission. The diagram below shows the components of the submission directory name.



Students can't normally copy files into this directory because it is owned by the professor. However the submit program has special execute permissions which set the user-id to root when this program is run. Having root permission then allows submit to copy files between users.

Your program is going to work the same way except that you don't get setuid-root permission and thus you cannot actually copy to the course directory. Instead as a base directory you should just use your current directory: "." Your program will ask the user for the course name, assignment name, and files to be submitted, and then copy those files.

Your program's output should be neatly formatted just like the real submit program. For example, submit shows a list of files like this:

```
 SIZE DATE   TIME  NAME
   50 Feb 28 23:26 Makefile
19498 Feb 28 23:26 mysubmit
28644 Feb 28 23:26 mysubmit.c
```

Preparation:
This submit program assumes that the course directory and the submit directory already exist. Within whatever directory you are going to test your program, make a pretend course directory (e.g. cos350) and within that make a submit directory.

Requirements:
1. Ask the user for the name of the course and verify that directory exists. (use stat())
2. Verify that it contains a submit directory. (use stat())
3. Get the user's name from the system. (use getuid() and getpwuid())
4. If this user does not yet have a subdirectory within the submit directory, create one. (use stat() and mkdir())
5. Ask the user for the name of the assignment, and create this subdirectory if it does not already exist. (use stat() and mkdir())
6. Show the user a list of the files in their current directory. These are the files that they may be submitting. The files should be listed alphabetically, and for each file show the size, modification date & time, and the file name. Don't show hidden files such as "." and ".." (use scandir() and stat())

(OVER)

7. Ask the user for the names of the files to submit. Allow * for everything. (You do not need to do any wildcard matching or copy directories.)
8. Copy the files into the submission directory. (You might want to look at the cp program from Lecture 7.)
9. Finally, show the user the files in the submission directory along with their sizes and modification times. (Hopefully you can just call the same function that you wrote for part 6.)
10. Create a Makefile for compiling your program. This may be very simple if you have only 1 source file.

Suggestions:

1. Look at the man pages for the various system calls (stat(2), getuid(2), getpwuid(3), mkdir(2), scandir(3)). E.g. man 2 stat
2. Always check the return values of library calls and print an error message if something goes wrong. *(This will help you debug your code if there is a problem.)*
3. Keep your program simple. You can make simplifying assumptions about the lengths of the various input pieces of the submission directory name.
4. Work in stages. Get small pieces working, and then gradually add more functionality. You might start with the function to list the files in a directory since this is the most important part.
5. Use the debugger to step through your program and see what is going on.
6. Some advanced students may already know how to invoke another unix program from within their program. You are NOT allowed to do this. (It also would be a very big security risk in a setuid program.)

What to turn in:

Written report:

- Your name(s) and, if a team, who did the electronic submission.
- A discussion of any incomplete parts, known bugs, deviations from the specification, or extra features in your program
- A listing of your code.
- Your Makefile
- A log of the following testing results. (I recommend using script to capture it.)

`mkdir cos101`	*make a new empty directory for testing*
`mkdir cos101/submit`	*make the submit subdirectory*
`mysubmit`	*run your program and submit to cos101 the three files: mysubmit.c mysubmit Makefile*
`ls –lR cos101`	*recursive listing to verify everything worked*
`mkdir cos102`	*make a new empty directory for testing*
`mkdir cos102/submit`	*make the submit subdirectory*
`mysubmit`	*run your program and submit to cos102, but this time just use * to submit everything*
`ls –lR cos102`	*recursive listing to verify everything worked*
`mysubmit`	*run it again but this time use a non-existent course directory cos103 to test error handling*

Electronic submission:

- Before you submit your program, compile and test it on a **Linux** machine in the lab.
- From a machine in the lab run the program "submit" to submit your files.
- Submit your source code (mysubmit.c). your executable (mysubmit), and your Makefile to a directory named: **prog3**
- **Do not in any way combine, compress, zip, or tar your files!**

Grading:

5 points: Ask Course Name & Verify
5 points: the submit dir exists
5 points: Get the user's name
5 points: Check for and create the user dir.
5 points: Ask & create the assignment dir.
25 points: Show the user the list of files to submit.
5 points: Ask for files to submit.
15 points: Copy the list of files.
10 points: Copy *
5 points: Show the final contents of the submission dir
5 points: Makefile
10 points: error handling