2 C Programming on a UNIX Operating System

In this class you will learn how to program in C on a UNIX operating system. The focus is on acquiring skills to be proficient in the development and execution of scientific and engineering computer software.

2.1 UNIX Operating System

Developed at AT&T Bell Labs in 1969.

Standard operating system used in scientific applications. Runs on virtually all of today's super computers and your Mac or Linux PC.

Many variants and descendants:

1. System V
   (a) Solaris
   (b) IBM's AIX
   (c) HP UX
2. BSD
   (a) FreeBSD
   (b) Mac OS X
3. GNU Project and Linux

UNIX is responsible for popularizing many features of modern operating systems:

1. Kernel abstracts all hardware related tasks
2. File system structure (subdirectory) and access model:
   $\Longrightarrow$ directories and text oriented access
3. The UNIX shell
4. Written in a high level language (C):
   $\Longrightarrow$ easy to port to different hardware

2.2 C Programming Language

Developed at AT&T Bell Labs in 1972 to help implement the UNIX operating system. Some characteristics of C are:

1. Low-level memory access (pointers)
   could be used as a "replacement" to assembly
2. Small set of keywords
3. High-level language constructions
   structured programming (based on procedures, functions — no jumps)
4. Complex functionality not implemented by the language
   (strings, I/O, math, network)

A first program in C:

```c
/* My first C program */
#include <stdio.h>

int main(int argc, char *argv[])
{
  printf("Hello world!\n");
  return 0;
}
```

2.3 Logging into a computer running UNIX

You will need:

1. computer name: `iacs5.ucsd.edu`
2. user name: for example `me9fxy` (substitute your own)
3. password

Type user name at the Login prompt and hit the `<return>` key.

Type one's password at the Password prompt and hit a `<return>` key.
(The password will NOT appear on the screen).

A typical screen output for the username me9fxy is

```
Login: me9fxy
Password:

Please wait...checking for disk quotas

We will be welcomed by iacs5:
iacs5.ucsd.edu%
```

The following is the remaining of this user's section. He/she sets the terminal type, changes the password and logs out.

```
iacs5.ucsd.edu% set term=vt100
iacs5.ucsd.edu% gpasswd
iacs5.ucsd.edu% logout
```

From a networked desktop, we use SSH (Secure Shell) to open an interactive login session on a multi-user system, such as `iacs5.ucsd.edu`. A typical session is as follows:

```
at.my.computer% ssh me9fxy@iacs5.ucsd.edu
me9fxy@iacs5.ucsd.edu's password:

Please wait...checking for disk quotas

We will be welcomed by iacs5:
iacs5.ucsd.edu%
```

2.4 Basic UNIX shell commands

Keep in mind that in UNIX *characters are all case-sensitive, including file names and passwords!*

UNIX shell commands often accept many options in the form of switches which modify their behavior or specify extra parameters.

On-line documentation is available on UNIX systems in the form of man pages. For example `man ls` displays documentation on the command `ls`.

If you are not sure about the exact name of the command use the switch `-k`. For example `man -k directory` will list all commands that have directory on their description.

Commands for directory and file manipulation

`pwd`	print the path name of the working directory
`cd`	change working directory
`ls`	list content of current directory
`cp`	copy files
`rm`	remove files
`rmdir`	remove empty directory
`mkdir`	create new directory
`cat`	prints files
`more`	paginates files (only forward)
`less`	paginates files (backward and forward)

Examples (The text following <- is not part of the section)

```
iacs5.ucsd.edu% pwd
/home/solaris/iacs5/me9f/me9fxy   <- Your home directory (~)

iacs5.ucsd.edu% cd ~/../public    <- Go to me9f/public/

iacs5.ucsd.edu% pwd
/home/solaris/iacs5/me9f/public

iacs5.ucsd.edu% ls                <- List files
cshrc         homework     locallogin   quiz         solution
exam          hw1.txt      login        readme.txt   textbook

iacs5.ucsd.edu% ls -F             <- Add / to directories
cshrc         homework/    locallogin   quiz/        solution/
exam/         hw1.txt      login        readme.txt   textbook/

iacs5.ucsd.edu% cd quiz           <- Go to me9f/public/quiz

iacs5.ucsd.edu% cd ~              <- Get back to me9f/me9fxy/

iacs5.ucsd.edu% pwd
/home/solaris/iacs5/me9f/me9fxy
```

Assigned homework, quiz, and exam problems will be stored in the public directory.
To copy the problems into your home directory, go to your home directory and use the cp command:

```
iacs5.ucsd.edu% cd ~
iacs5.ucsd.edu% cp ~/../public/homework/hw1.txt .
```

```
iacs5.ucsd.edu% cp ~/../public/quiz/quiz1.txt .
```

Alternatively, go to the homework or quiz directory and copy from there

```
iacs5.ucsd.edu% cd ~/../public/homework
iacs5.ucsd.edu% cp hw1.txt ~/.
iacs5.ucsd.edu% cd ~/../public/quiz
iacs5.ucsd.edu% cp quiz1.txt ~/.
```

This creates a directory in the user's home directory and copies multiple files

```
iacs5.ucsd.edu% cd ~
iacs5.ucsd.edu% mkdir hm4
iacs5.ucsd.edu% cp ~/../public/hw4.txt ~/../public/hw4.data hm4
iacs5.ucsd.edu% cd hm4
iacs5.ucsd.edu% ls
hw4.data     hw4.txt
iacs5.ucsd.edu% less hw4.txt
```

The `less` command displays the contents of file `hw4.txt`.

Use the switch `-r` (recursive) to engage all files and subdirectories in a command. For example, the following session copies all files and subdirectories under directory `old` to a new directory `new` and removes the `old` directory:

```
iacs5.ucsd.edu% mkdir new
iacs5.ucsd.edu% cp -r old new
iacs5.ucsd.edu% rm -r old
```

2.5 Compiling and running your first program

Use your favorite editor `vi` or `emacs` to create a source program, say `first.c`, and type the following lines:

```c
/* My first C program */
#include <stdio.h>

int main(int argc, char *argv[])
{
  printf("Hello world!\n");
  return 0;
}
```

The following commands compiles and links the program `first.c` located in your current directory:

```
iacs5.ucsd.edu% ls
first.c
```

```
iacs5.ucsd.edu% gcc first.c
iacs5.ucsd.edu% ls
a.out          first.c
```

creating an executable program named `a.out` also in the current directory. The following commands run the program:

```
iacs5.ucsd.edu% ./a.out
Hello world!
iacs5.ucsd.edu%
```

You can assign a name other than `a.out` to the executable file name by using the switch `-o` as follows:

```
iacs5.ucsd.edu% rm a.out
iacs5.ucsd.edu% ls
first.c
iacs5.ucsd.edu% gcc -o any_name first.c
iacs5.ucsd.edu% ls
any_name       first.c
iacs5.ucsd.edu% ./any_name
```

The last instruction executes the program `any_name`.