

First:

Login to your Unix account and type:

% `/home/cs476/mlist`

To put your login name in cs476 mailing list.

Introduction to UNIX

→ Dot files:

To change the behavior of programs.

Examples: `.login, .cshrc, .mailrc, .forward, etc...`

→ Unix Man pages:

Extensive on-line documentation.

Examples: `man, info, apropos, whereis, which, etc..`

→ Editing:

You may use any program to edit a file.

Examples: `vi, emacs, ed, MS word, notepad, etc...`

→ Program Management:

❖ Makefile:

```
hello : hello.o PrintDate.o
        <tab> gcc -o hello hello.o PrintDate.o
hello.o : hello.c
        <tab> gcc -c hello.c
PrintDate.o : PrintDate.c
        <tab> gcc -c PrintDate.c
clean:
        <tab> rm *.o hello
```

hello.c:

```
main()
{
    printf("Hello World\n");
    PrintDate();
}
```

PrintDate.c:

```
void PrintDate()
{
    system("date");
}
```

```
% make          /* produce hello */
% make clean    /* removes *.o and hello */
```

❖ Archive:

```
% tar cf hello.tar * /* create */
% tar tf hello.tar   /* list */
% tar xf hello.tar   /* extract */

% gzip hello.tar     /* compress */
% gunzip hello.tar.gz /* uncompress */
```

❖ Libraries:

PrintDate.c:

```
void PrintDate()
{
```

```

        system("date");
    }
Math.c:
char *compare(int i1, int i2)
{
    if (i1 > i2) return "first";
    else if (i2 > i1) return "second";
    else return "equal";
}
int diff (int i1, int i2)
{
    return (i1-i2);
}
int sum (int i1, int i2)
{
    return (i1+i2);
}

```

Makefile:

```

OBJ= PrintDate.o Math.o
all: $(OBJ) libcs476.a
PrintDate.o : PrintDate.c
            gcc -c PrintDate.c
Math.o : Math.c
            gcc -c Math.c
libcs476.a : $(OBJ)
            ar r libcs476.a $(OBJ)
clean:
            rm *.o libcs476.a

```

libcs476.h:

```

void PrintDate();
char *compare(int, int);
int sum(int, int);

```

To list the contents of the library:

```
% ar t libcs476.a
```

Example to use the library:

testlibcs476.c:

```

#include <stdio.h>
#include "libcs476.h"
int main()
{
    PrintDate();
    printf("%s\n",compare(13, 27));
    printf("%d\n", diff (13, 27));
    printf("%d\n", sum (13, 27));
}

```

Makefile:

```

testlibcs476 : testlibcs476.c
    gcc -o testlibcs476 testlibcs476.c
-lcs476 -L.
clean:
    rm testlibcs476

```

→ Unix file system:

```

% ls -l
-rwx rwx rwx  2  wahab faculty 1278 Aug 25 file1

```

```

% touch

```

Example:

```

% mkdir test
% cd test

% touch file
% ls -l
-rw----- 1 wahab faculty 0 Aug 1 23:25 file
% /usr/bin/touch 1226112272 file
% ls -l
-rw----- 1 wahab faculty 0 Dec 26 1972 file
% stat file
/* all details about the file */
.....
Modify: 1972-12-26 11:22:00

```

.....

% **find**

Example :

Remove all files under directory test named
a.out or *.o that have not been accessed for a week:

```
% find ./test \( -name a.out -o -name '*.o' \)  
-atime +7 -exec rm {} \;
```

% **chmod -R** a+rx *

% **cp -r** path1 path2

% **rm -r** path

→ The cat & tail story:

Creating the world's first chat!

```
% cat >> /tmp/wahab  
% tail -f /tmp/Wahab
```

→ Which Shell?

```
% sh (Bourne shell, the oldest shell)  
% bash (Bourne-again shell, a super set of sh)  
% csh (Berkeley, syntax resembles C language)  
% tcsh (derivative of csh)  
% ksh (korn shell-the newest shell)
```

→ All Shells has:

| > >> < &

E.g. % **ls -lt | more**

Example 1: Producing pdf version of a man page:

% **whereis touch**

touch: /usr/bin/touch /usr/ucb/touch /usr/local/bin/touch /usr/man/man1/touch.1
/usr/man/man1b/touch.1b

% **troff -man /usr/man/man1/touch.1 > t1**

% **dpost t1 > t2**

% **ps2pdf t2 touch.pdf**

OR combine all the above commands using | as:

% **troff -man /usr/man/man1/touch.1 | dpost | ps2pdf - touch.pdf**

Example 2:

% **cb <prog.c > prog.b**

% **more prog.b**

% **mv prog.b prog.c**

What will happen when you do:

% **cb <prog.c > prog.c ???**

We may combine all the above commands using | as:

% **cat prog.c | cb | tee prog.b**

You can use instead of cb:

% **indent prog.c**

Example 3:

% **echo hi > tmp**

% **tail -f tmp >> tmp &**

% **tail -f tmp**

→ Some Shells (e.g. csh) has:

Job control

% **sort** *bigfile1* > *bigfile2*

^Z

% **jobs**

..... list of background jobs

% **kill -9 %job**

% **ps**

..... list of processes

% **kill -9 process**

We can also use:

% **pgrep** <name>

% **pkill** <name>

→ Quoting:

|C , |.....| , |.....| , "....."

% echo *

% echo |* *quote ONE char*

% echo |***>?| *quote ANY number of chars*

% mail |cat list| < letter *command substitution*

% echo " |pwd| \$home |" *does not quote | ... | and \$*



Shell and Environment Variables:

bourne shell:

```
$ path=./bin:/usr/bin  path is a standard shell variable
$ echo $path
$ d=`pwd`              d is a user defined shell variable
$ echo $d
```

Note: there should be **no spaces** around **=**

csH shell:

```
% set path = ./bin:/usr/bin
% set d = `pwd`
% echo $d

% setenv DISPLAY dogwood.cs.odu.edu:0
% echo $DISPLAY
```

Environment variables:

are **automatically** exported to all child processes.

Shell variables:

has to be **explicitly** exported.

Example:

```
% set x=`pwd`
% setenv y `pwd`
% echo $x $y
```

```
% csh
% echo $x $y x is undefined but y is defined.
```

→ **Bit-level operations:**

You may access individual bits of any byte.

Example: bits

```
char B=0;
int i;

B=B | (1<<4); /*set 4th bit*/

printf("%d ", (B&(1<<i))>>i); /* print ith bit */

B=B & ~(1<<4); /*clear 4th bit*/
```

→ **Debugging:**

You may use any debugger to debug a program.

Example: gdb

Using `-g` flag to debug prog:

```
% gcc -g -o prog prog.c
```

Some gdb commands:

run, list, break <n>, clear <n>, display <var>, next,
continue, quit