Chapter 8

Libraries

Libraries

Libraries are sets of related functions packaged as a system resource for use by other programs.

Libraries may contain functions which handle file systems, I/O, math, networking, graphics, debugging, etc...

Libraries

Some libraries are standard such as the standard C libraries.

Libraries may be created by individuals or large companies to support a product.

They may be free, or may be very costly.

Advantages of Libraries

Libraries are generally written by experts so their code is typically efficient.

Libraries are generally used and tested by many programmers so they are typically "bug-free".

- **They can contain code which is commonly used (such as *string* functions).**
- **They can contain code which is difficult to write (such as *math* functions).**
- **They can contain code which is hardware dependent (such as *graphics* functions).**

Libraries

A program may use a single function from a single library. Or it may use many functions from many libraries.

Libraries may be built “on top” of another library.

On library may use functions from other libraries which in turn use functions from still other libraries.

Using Libraries

There are two steps in using a library.

1) The library header file must be included so that the code "knows" how to use the functions contained in the library.

```
#include <math.h>
#include <complex.h>

int main(void)
{ complex z = 1 + I*1;
  printf("z = %f + j%f\n",
                creal(z), cimag(z));
```

Using Libraries

2) The library must be linked when creating the executable file.

```
gcc include.c –o include -lm
```

The switch `-l` tells gcc to include a certain library module when linking. The `m` states to include the module which contains both `complex.h` and `math.h`.

Header Files

Header files contain the "directions" for using the code of the associated library.

- **Contains function prototypes describing the parameters and return value of a function.**

```
long double powl (long double, long double);
```

- **Contains constants relative to the library.**

```
#define M_PI_4  0.78539816339744830962
```

- **Contains macro definitions.**

```
#define stdin  (&_iob[STDIN_FILENO])
#define isnormal(x) (fpclassify(x)==FP_NORMAL)
```

Header Files

- Contains `typedefs`.

```
typedef struct _iobuf
   {  char* _ptr;
      int   _cnt;
      char* _base;
      int   _flag;
      int   _file;
      int   _charbuf;
      int   _bufsiz;
      char* _tmpfname;
   } FILE;
```

- Contains external variable declarations .

```
extern int errno;
```

Library Files

On most linux systems, library files are store in the `/usr/lib` directory.

Library files begin with `lib` and end with the suffix `.a`. For the math library mentioned earlier, the name is `libm.a`.

```
[breid@ullab111 ~]$ ls -l /usr/lib/lib*.a
-rw-r--r-- 1 root root 313430 Jan 24  2003 /usr/lib/liba2ps.a
-rw-r--r-- 1 root root  51524 Jan 28  2003 /usr/lib/libacl.a
-rw-r--r-- 1 root root   9706 Sep 14  2004 /usr/lib/libaio.a
-rw-r--r-- 1 root root  10964 Feb 23  2005 /usr/lib/libanl.a
-rw-r--r-- 1 root root  12090 Feb 23  2005 /usr/lib/libanl_p.a
-rw-r--r-- 1 root root   7346 Feb 24  2003 /usr/lib/libapm.a
-rw-r--r-- 1 root root   5386 Jan 28  2003 /usr/lib/libattr.a
```

Library Files

On a Microsoft Windows system, the libraries may be located `windows\system`, `windows\system32`, or the `\lib` directory of a compiler.

MS Library files end with the suffix `.lib` or `.dll`.

```
c:\>dir c:\Program Files\CBuilder5\Lib *.lib
01/31/2000  05:00 AM           130,048 bcbatl.lib
08/07/2000  05:01 AM           483,828 bcbie50.lib
01/31/2000  05:00 AM         2,371,204 bcbsmp50.lib
01/31/2000  05:00 AM         2,096,128 bfc42.lib
01/31/2000  05:00 AM         2,502,144 bfc42d.lib
01/31/2000  05:00 AM            50,176 bfcs42.lib
01/31/2000  05:00 AM            50,176 bfcs42d.lib
01/31/2000  05:00 AM           200,704 cg32.lib
```

Linking Libraries

If a Library is contained in a directory other than the `/usr/lib` directory or any directory listed in the compiler's path, the `-L` switch can be used to expand the compiler's search path.

For example, the X Library is commonly stored in `/usr/X11R6` so we would use the following command:

```
gcc proj.c -lX11 -L/usr/X11R6/lib
```

C Standard Library

The "C Standard Library" is actually a collection of libraries which contain the most commonly used functions of C such as `printf()`, `strcpy()`, `fopen()`, and `sqrt()`.

Most compilers link to the core of the C Standard Library by default. Thus, the following two lines are equivalnet:

```
gcc proj.c –o proj.exe
gcc proj.c –o proj.exe -lc
```

http://en.wikipedia.org/wiki/C_standard_library

Failure to Include C Standard Library Headers

Because the C Standard Library is usually linked by default, not including appropriate header files can cause interesting (i.e. incorrect) results. The following code may produce unexpected results on some computers:

```
int main(void)
{ double x, y;
  x = 2;
  y = sqrt(x);
  printf("sqrt(%lf) = %lf\n", x, y);
}
```

Curses Library

The `curses` Library is a graphics library for use on a character terminal screen.

Even in the day of high-end graphics, a low-level graphics library is important for displays prior to loading an O/S or for primitive O/S's on embedded systems.

Development files may not be included in basic OS install. Add development library `libncurses5-dev` using Ubuntu Software Center or

```
sudo apt-get install libncurses5-dev
```

Curses Library

Compile with: `gcc hello1.c -lncurses`

```
#include <curses.h>
int main(void)
{  initscr();
   clear();
   move(5,10);
   addstr("Clemson");
   move(LINES-1, 0);
   refresh();
   getch();
   endwin();
```

Must call once.

- Sets global variables like LINES and COLS
- Takes over `stdin` and `stdout` (but not `stderr`)

Call to clear and reset terminal to initial settings.
Can now use `printf()`, `fgets()`, and other functions on `stdin` and `stdout`

Bug in code? In terminal type: `stty sane`

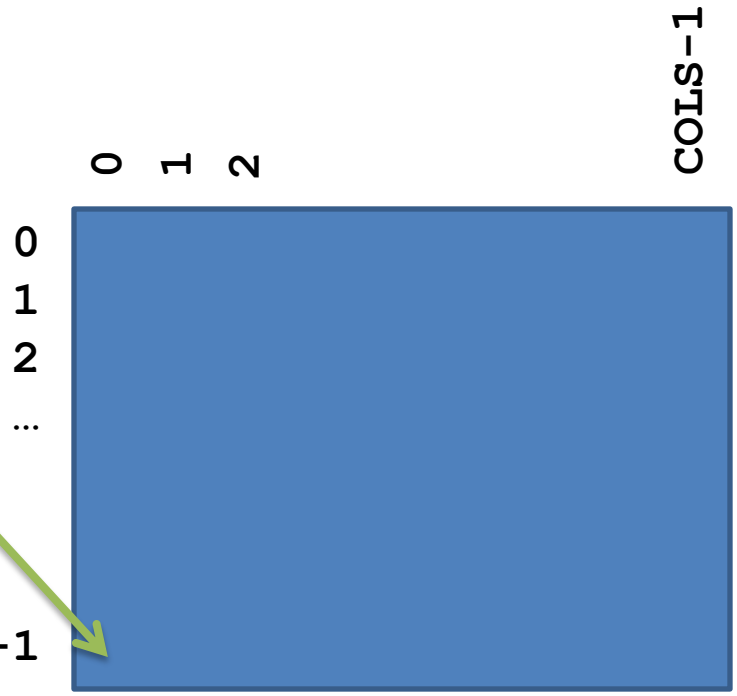
Curses Library

```
#include <curses.h>
int main(void)
{ initscr();
  clear();
  move(5,10);
  addstr("Clemson");
  move(LINES-1, 0);
  refresh();
  getch();
  endwin();
```

Clear the screen

Move cursor to position (line, column)

Add string to buffer and advance cursor



Curses Library

```
#include <curses.h>
int main(void)
{ initscr();
  clear();
  move(5,10);
  addstr("Clemson");
  move(LINES-1, 0);
  refresh();
  getch();
  endwin();
```

Other routines merely manipulate data structures

Flush output to the terminal

Block waiting for user to type anything

Change Text Appearance

hello3.c

```
#include <curses.h>
int main(void)
{ initscr();
  clear();
  for(i=0; i<LINES; i++) {
    move(i,i+1);
    if (i%2==1)
      standout();
    addstr("Hello world");
    if (i%2==1)
      standend();
  }
  refresh();
  getch();
  endwin();
```

Toggle attribute

Applies to all subsequent characters written to window

End of this attribute property

Many other attributes available (e.g., color)

Can Add Timing

hello4.c

```
#include <curses.h>
int main(void)
{ initscr();
  clear();
  for(i=0; i<LINES; i++) {
    move(i,i+1);
    if (i%2==1)
      standout();
    addstr("Hello world");
    if (i%2==1)
      standend();
    sleep(1)
    refresh();
  }
  getch();
  endwin();
```

Create Illusion of Moving

hello5.c

```
#include <curses.h>
int main(void)
{ initscr();
  clear();
  for(i=0; i<LINES; i++) {
    move(i,i+1);
    addstr("Hello world");
    refresh();
    usleep(100000);
    move(i,i+1);
    addstr("                ");
  }
  getch();
  endwin();
```

Sleep in μ seconds

Move back to start of string

Another example with moving at different rates **vartiming.c**

`curses.h` Functions

The `curses` Library contains many, many functions for the console. Needed for MP:

- `initscr()` – **Turn on curses functionality.**
- `endwin()` – **Turn off curses functionality.**
- `clear()` – **Clear the Screen.**
- `move()` – **Move the cursor.**
- `addstr()` – **Print a string.**
- `refresh()` – **Flush the display buffer.**
- `getch()` – **Get a character from the keyboard.**
- `standout()standend()` – **Toggle attribute.**

curses.h Functions 2

More curses functions for animation effects:

- nocbreak() – Turn on line buffering for input.
- cbreak() – Turn off line buffering for input.

  hello2.c mover1.c
- noecho() – Turn off echoing of input to output.
- echo() – Turn on echoing of input to output.

- nodelay() – Don't wait for input.
- delay() – Wait for input.

`curses.h` Functions 3

Still more `curses` functions:

- **`halfdelay(n)` – Block for `n`/10 of a second.**
- **`usleep(n)` – Sleep for `n` microseconds.**

`mover2.c` `pacman.c`

- **`getyx(stdscr, y, x)` – Macro to place the current cursor position in the two integer variables y and x.**
- **`keypad(stdscr, TRUE)` – Enable keypad for backspace, arrow, and function keys. See also curses.h**