

Announcements

- Program #3
 - Is on the web
- Reading
 - Chapter 16, 13.3 (Today)
 - Skip 16.4-16.6 and 16.9
 - Chapter 17 (Thursday)

Additional Standard Library Functions

- Random Numbers
 - in **<stdlib.h>**
 - void srand(unsigned int);
 - seed the random number generator
 - int rand(void);
 - return a pseudo-random number between 0 and RAND_MAX
- Floating Point (all use doubles)
 - prototypes defined in **<math.h>**
 - compile using **-lm** option
 - double sqrt(double value);
 - compute square root of value
 - Bad parameters produce domain errors
 - errno is set to EDOM
 - double exp(double value) and double exp10(double value)
 - computes e^{value} and 2^{value}

Trig Functions

- `double sin(double angle);`
 - computes sin of angle (in radians)
- Also have:
 - `double cos(double angle);`
 - `double tan(double angle);`
 - `double asin(double value);`
 - `double acos(double value);`
 - `double atan(double value);`

Other Handy Functions

- Fractions

- `double floor(double x);`
 - next lowest whole integer
- `double ceil(double x);`
 - next highest whole integer
- `double fabs(double x);`
 - absolute value
- `double fmod(double x, double y);`
 - restricts `y` to an integer

Function Pointers

- Pointers can also be of type function
 - `void (*myVoidFunc)(int);`
 - declare a variable `myVoidFunc` which points to a function which takes an integer as a parameter.
 - Like any pointer variable, declaring it does not create an instance of what it points to!
- Uses
 - Create Object Oriented Code
 - Callbacks from utility routines
- Often a good idea to typedef each function pointer
 - `typedef void(*myVoidFuncPtr)(int);`

Function Pointer Example

```
union myDataType {  
    int a;  
    float b;  
} ;
```

```
typedef void (*myPrintFuncPtr)(union myDataType);
```

```
typedef struct {  
    myPrintFuncPtr printIt;  
    union myDataType data;  
} myObject ;
```

```
void printInt(union myDataType data) {  
    printf("data = %d\n", data.a);  
}
```

Function Pointer Example Continued

```
int main(void) {  
    int i;  
    myObject *objects;  
  
    objects = (myObject *) calloc(sizeof(myObject), 5);  
  
    objects[0].data.a = 43;  
    objects[0].printIt = printInt;  
    objects[1].data.b = 3.1415;  
    objects[1].printIt = printFloat;  
  
    for (i=0; i < 2; i++) {  
        objects[i].printIt(objects[i].data);  
    }  
    exit(0);  
}
```

Callback Example

```
typedef int (*compareFunc)(item *a, item * b);

int searchTree(Node *root, item *target, compareFunc cmp) {
    if (!root) return -1;
    ret = (cmp)(target, root->data);
    if (ret == 0) {
        /* found it */
        return 1;
    } else if (ret < 0) {
        return searchTree(root->left, target, cmp);
    } else {
        return searchTree(root->right, target, cmp);
    }
}
```

Date & Time Functions

- `clock_t clock(void);`
 - process time since start of program execution
 - to convert to time, use `CLOCKS_PER_SEC`
- `time_t time(time_t *val);`
 - fill `val` with the current time (in machine dependent format)
- `char *ctime(time_t *val);`
 - return a character representation of the passed time
 - Sun Jul 4 04:02:48 2005\n\0
- `double difftime(time_t time1, time_t time2)`
 - return number of seconds between `time1` and `time2`
- `struct tm *gmtime(time_t val)`
`struct tm*localtime(time_t val)`
 - convert to UTC or local time

Execution Environment

- Program Termination

- abort(void);
 - terminate program with error (usually a core time)
- atexit(void (func)(void));
 - on termination call function func
- exit(int status);

- Running Shell Commands

- void system(char *command);
 - runs command (not all systems support it)

- Sorting

- void qsort(void *base, size_t number, size_t elementSize, int (*compare)(void const *, void const *));
 - Sort the passed array, using passed compare function
 - strcmp will work for this!

Creating your own libraries

- Libraries
 - collections of object (.o) files that work together
 - can be linked into other programs
 - linking can happen prior to execution
 - linking can be done during program execution
- Static libraries
 - linked into the program as part of the link step
 - require space in each executable that uses them
- Dynamic (shared) libraries
 - linked into the program at program startup
 - require only one copy for the entire system
 - Have version numbers associated with them
 - controls which version work with which applications

Creating Static Libraries

- UNIX Command `ar`
 - `LIBRARY=myLib.a`
 - `OBJS=file1.o file2.o file3.o`
 - `ar cru $(LIBRARY) $(OBJS)`
- Using a Static Library
 - `gcc -o myProgram main.o myLib.a`

Creating Dynamic (shared) Libraries

- Use special gcc flags
 - -nostdlib -shared -fPIC -Wl,-soname,mylib.so.1
 - -nostdlib - no standard C library needed
 - -shared - generate a shared library
 - -fPIC - generate position independent code
 - -Wl,-soname,mylib.so.1
 - name the shared object mylib.so.1
- Example Makefile Rules

```
LIBFLAGS = -nostdlib -shared -fPIC -Wl,-soname,$@.1

libavl.so:  avl.c table.h
    $(CC) $(LIBFLAGS) $(CFLAGS) avl.c -o libavl.so
ln -f -s libavl.so libavl.so.1
```

Runtime Linking

- Can Load a library at runtime
 - Allows skins, plugins, etc to work
- C functions to support this:
 - `void * dlopen(const char *pathname, int mode)`
 - `pathname` is the name of a shared library
 - `mode` controls operation (`RTLD_NOW`)
 - `void *dlsym(void *handle, const char *name);`
 - lookup a function by name in the passed shared library
 - Return a pointer to that function (or `NULL` if not found)
 - `int dlclose(void *handle);`
 - returns 0 on success

Dlopen Example

```
typedef void (*displayFuncPtr)(char *file);
typedef struct {
    displayFuncPtr displayFunc;  char *mediaType;
} renderTypePlugin;

renderTypePlugin *loadPlugin(char *library, char *mediaType) {
    void *dIPtr;  renderTypePlugin *plugin;
    dIPtr = dlopen(library, RTLD_NOW);
    if (!ptr) return NULL;
    /* ... allocate plugin and check return ... */
    plugin->displayFunc = (displayFuncPtr ) dlsym(dIPtr, "draw");
    if (!plugin->displayFunc)  /* error, couldn't find function */
    plugin->mediaType = strdup(mediaType);
    dlclose(dIPtr);
    return plugin;
}
```