CSC 4304 - Systems Programming
Fall 2008

LECTURE - VI

FILES & DIRECTORIES

Tevfik Koşar

Louisiana State University

September 18th, 2008

Summary of Last Class

- Buffered File I/O
  - opening and closing streams
  - reading from / writing to streams
  - Binary I/O
  - Formatted I/O

In Today's Class

- More C Fundamentals
- Files and Directories
- Project 1 out

Pointer to Structures

```
struct rec{
        int x;
        …
};
struct rec r, *rp = &r;
  r.x
  rp->x
---
void f(struct rec *rp);
f(&r);
---
struct tnode{
        int data;
        struct tnode *left, *right;
}
```

Function Pointers

- Functions are not variables but we can define pointers to functions which will allow us to manipulate functions like variables..

- int f() : a function which returns an integer
- int* f() : a function which returns a pointer to integer
- int (*f)(): a pointer to a function which returns integer
- int (*f[])(): an array of pointer to a function which returns integer

Example

```
   void sum(int a, int b) {printf("sum: %d\n", a+b);}
   void dif(int a, int b) {printf("dif: %d\n", a-b);}
   void mul(int a, int b) {printf("mul: %d\n", a*b);}
   void div(int a, int b) {printf("div: %f\n", a/b);}

   void (*p[4]) (int x, int y);

int main(void)
{
  int result;
  int i=10, j=5, op;

  p[0] = sum; /* address of sum() */
  p[1] = dif; /* address of dif() */
  p[2] = mul; /* address of mul() */
  p[3] = div; /* address of div() */

  for (op=0;op<4;op++) (*p[op]) (i, j);

}
```

Operator Precedence

Operators	Associativity	Type
`()` `[]` `->` `.`	left to right	primary expr.
`++` *(postfix)* `--` *(postfix)*	right to left	postfix
`+` `-` `!` `++` *(prefix)* `--` *(prefix)* (*type*)	right to left	unary
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`&&`	left to right	logical AND
`||`	left to right	logical OR
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment
`,`	left to right	comma

Complicated Declarations

- int *a[10]: array[10] of pointer to int
- int (*a)[10]: pointer to array[10] of int
- void *f(): function returning pointer to void
- void (*f)(): pointer to a function returning void
- char (*(*x())[])(): function returning pointer to array[] of pointer to function returning char
- char (*(*x[3])())[5]: array[3] of pointer to function returning pointer to array[5] of char

Files and Directories

- Objectives
  - Additional Features of the File System
  - Properties of a File.

```
struct stat {
    mode_t    st_mode; /* type & mode */
    ino_t     st_ino; /* i-node number */
    dev_t     st_dev; /* device no (filesystem) */
    dev_t     st_rdev; /* device no for special file */
    nlink_t   st_nlink; /* # of links */
    uid_t     st_uid;        gid_t     st_gid;
    off_t     st_size; /* sizes in byes */
    time_t    st_atime; /* last access time */
    time_t    st_mtime; /* last modification time */
    time_t    st_ctime; /* time for last status change */
    long      st_blk_size; /* best I/O block size */
    long      st_blocks; /* number of 512-byte blocks allocated */
};
```


Stat Functions

- Three major functions:

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *pathname, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *pathname, struct stat *buf);
```

- ☑ stat returns info about a named file
- ☑ fstat returns info about an already open file
- ☑ lstat returns info about a symbolic link, not the referenced file

Directories

- dirent : file system independent directory entry

```
struct dirent{
      ino_t  d_ino;
      char   d_name[];
       ....
};
```


Example 1: Simple ls

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>

do_ls(char dirname[])
{
      DIR *dir_ptr;
      struct dirent *direntp;

      if ((dir_ptr = opendir(dirname)) == NULL)
            printf("error!\n");
      else{
            while((direntp = readdir(dir_ptr)) != NULL)
                  printf("%s\n", direntp->d_name);
            closedir(dir_ptr);
      }
}

main(int argc, char* argv[]){
      if (argc == 1)   do_ls(".");
      else             do_ls(argv[1]);
}
```

Example 2: Get File Info

```
show_stat_info(char *fname, struct stat *buf)
{
        printf("   mode: %o\n", buf->st_mode);
        printf("  links: %d\n", buf->st_nlink);
        printf("   user: %d\n", buf->st_uid);
        printf("  group: %d\n", buf->st_gid);
        printf("   size: %d\n", buf->st_size);
        printf("modtime: %d\n", buf->st_mtime);
        printf("   name: %s\n", fname);

}
main(int argc, char* argv[]){
        struct stat info;

        if (argc > 1)
                if (stat(argv[1], &info) != -1){
                        show_stat_info(argv[1], &info);
                }

}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
```


Example 3: Get More Info

```
show_stat_info(char *fname, struct stat *buf)
{
        struct passwd *user = getpwuid(buf->st_uid);
        struct group *gr = getgrgid(buf->st_gid);

        printf("   mode: %o\n", buf->st_mode);
        printf("  links: %d\n", buf->st_nlink);
        printf("   user: %s\n", user->pw_name);
        printf("  group: %s\n", gr->gr_name);
        printf("   size: %d\n", buf->st_size);
        printf("modtime: %d\n", buf->st_mtime);
        printf("   name: %s\n", fname);

}
main(int argc, char* argv[]){
        struct stat info;

        if (argc > 1)
                if (stat(argv[1], &info) != -1){
                        show_stat_info(argv[1], &info);
                }

}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>
```

Summary

- More C Fundamentals
- Files and Directories
- Project 1 out

- Next Class: Continue Files and Directories

- Project 1 due October 2nd
- Read Ch 4 from Stevens..

Acknowledgments

- Advanced Programming in the Unix Environment by R. Stevens
- The C Programming Language by B. Kernighan and D. Ritchie
- Understanding Unix/Linux Programming by B. Molay
- Lecture notes from B. Molay (Harvard), T. Kuo (UT-Austin), G. Pierre (Vrije), M. Matthews (SC), and B. Knicki (WPI).