Module 5

Directories

Overview

- Objectives
- Relevance

Directory Format

```
1   #include <sys/stat.h>
2
3   main (int argc, char *argv[]){
4     struct stat buf;
5     if (stat(argv[1], &buf) != -1) {
6       if (S_ISDIR(buf.st_mode)) {
7       printf("%s is a directory.\n", argv[1]);
8       }
9     }
10  }
```


Accessing a Directory

- `opendir()`— Opens directory
- `readdir()`— Reads directory
- `closedir()`— Closes directory
- `telldir()`— Obtains current location associated with directory stream
- `seekdir()`— Sets pointer to next `readdr` operation
- `rewinddir()`— Resets position of directory stream

Accessing a Directory

```
1   #include <stdio.h>
2   #include <sys/stat.h>
3   #include <dirent.h>
4
5   int main(int argc, char *argv[]) {
6
7     struct stat buf;
8     DIR   *dirp;
9     struct dirent *dent;
10
11    if (argc < 2) {
12      fprintf(stderr, "Usage: %s directoryname\n",argv[0]);
13      exit(1);
14    }
15
16    if (stat(argv[1], &buf) == -1) {
17      perror("stat");
18      exit(1);
19    }
```



```
20
21    if (S_ISDIR(buf.st_mode)) {
22      printf("%s is a directory.\n", argv[1]);
23    } else {
24      fprintf(stderr, "%s is not a directory.\n", argv[1]);
25      exit(1);
26    }
27
28    if ((dirp = opendir(argv[1])) == NULL) {
29      perror("opendir");
30  exit(1);
31    }
32    printf("Contents\n========\n");
33
34    while (dent = readdir(dirp)) {
35      printf("%s\n", dent->d_name);
36    }
37    (void)closedir(dirp);
38  }
```


Using Hard and Symbolic Links

- `link()`— Creates a link
- `symlink()`— Creates a symbolic link
- `readlink()`— Reads value of a symbolic link

`mylinks.c`

```
1   #include <unistd.h>
2
3   main() {
4
5     /* Create a hard link */
6     if link("path1", "path2");
7       perror("link")
8   }
9
10    /* Create a symbolic link */
11    if link("path1", "path2");
12      perror("symlink")
13    }
14  }
```

myreadlink.c

```
1   #include <unistd.h>
2   #include <stdio.h>
3   #include <sys/stat.h>
4   #include <sys/types.h>
5   #include <stdlib.h>
6
7   int main(int argc, char *argv[]) {
8
9     char *buf;
10    struct stat statbuf;
11    int n;
12
13    if (argc < 2) {
14      fprintf(stderr, "Usage: %s linkname\n", argv[0]);
15      exit(1);
16  }
17
18  if (lstat(argv[1], &statbuf) == -1) {
19      perror("lstat");
20      exit(1);
```



```
21    }
22
23    if (!S_ISLNK(statbuf.st_mode)) {
24      fprintf(stderr, "%s is not a symbolic link.\n",
25          argv[1]);
26  exit(1);
27    }
28
29    buf = (char *)malloc(statbuf.st_size + 1);
30    if (buf == NULL) {
31      fprintf(stderr, "Out of memory.\n");
32      exit(1);
33    }
34
```



```
35    n = readlink(argv[1], buf, statbuf.st_size + 1);
36    if (n == -1) {
37      perror("readlink");
38      exit(1);
39    }
40    buf[n] = '\0';
41    printf("%s\n", buf);
42    exit(0);
43  }
```


Exercise: Directories

- Objectives
- Tasks
- Discussion
- Solutions