Directory based system calls

Directory System Calls

- getcwd()
- mkdir()
- chdir()
- rmdir()
- opendir()
- closedir()
- readdir()

getcwd() system Call

- To get the current working directory in UNIX, we use the “pwd” command.
- The system call behind the “pwd” command is the getcwd() call.

```
# include<unistd.h>
char *getcwd(char *buf,  size_t bufsize);
char *buf       /* Returned pathname */
size_t   bufsize       /* sizeof buf */
```

Returns current working directory on success and NULL on error

bufsize should the maximum size of path.

A program that uses the getcwd()

```
#include <stdio.h> #include <stdlib.h>#include <unistd.h>#include <limits.h>
#define   PATH_MAX   255
int main (void)     {
char dirname[PATH_MAX+1]; /* To be passed to getcwd system call. */
/* Use getcwd to get the name of the current working directory. */
if (getcwd(dirname, PATH_MAX) == NULL)      {
fprintf(stderr, "Could not obtain current working directory.\n");
exit(1);
}
else {
printf("Current working directory: %s\n", dirname);
}
return 0;
}
```

Current working directory: /home/wlab/Desktop/test

chdir () [cd command in Unix]

- # include<unistd.h>
  int chdir(const char *path);
- On success, zero is returned.
- On error, -1 is returned, and *errno* is set appropriately.
- This system call changes the current working directory to that specified in “path”.
- ERRORS: Depending on the file system, other errors can be returned. The more general errors for **chdir**() are listed below:

Error code	Description
EIO	**An I/O error occurred.**
ENAMETOOLONG	**path is too long.**
ENOENT	**The file does not exist.**

A program that uses the chdir()

```
#include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <errno.h>
const char * constpath = "/home/wlab/Download/test1";
int main ()
{
printf ("\n\n Changing directory to < %s >\n\n", path);
if (chdir (path) == -1)
{
printf ("\n\nchdir failed - %s\n\n", strerror (errno));  }
else
{
printf ("\n chdir done !!!\n");
printf ("\n\n directory content of %s\n\n", path);
system ("ls -l");
}
return 0;
}
```

mkdir() System Call

- # include<sys/stat.h>
- int mkdir(const char *path, mode_t perms)

  const char *path, /* Pathname */

  mode_t perms /* file access permissions */
- Upon successful completion, *mkdir*() shall return 0.
- Otherwise, -1 shall be returned, no directory shall be created, and *errno* shall be set to indicate the error.
- This function creates a new, empty directory.

Remove dir System Call

- An empty directory is deleted with rmdir() system call.
- # include<sys/stat.h>

  int rmdir(const char *path);

  char *path /* Pathname */
- It Returns 0 on success and -1 on error

Program to make and remove directory

```
#include <dirent.h>#include <stdio.h>#include <unistd.h>#include <stdlib.h>
#include <errno.h>
int main(int  argc, char *argv[])
{           int md, rd;
            DIR *ds;
            struct  dirent *dir;
            md = mkdir(argv[1],  0777);
            if(md == 0)
                        printf("%s  directory is created\n",  argv[1]);
            else
                        printf("%s  directory is not created\n", argv[1]);
            rd = rmdir(argv[2]);
            if(rd == 0)
                        printf("%s directory is removed\n", argv[2]);
            else
                        printf("%s directory is not removed\n", argv[2]);
}
```

opendir and closedir System Calls

- # include<dirent.h>

  DIR* opendir(const char *path);

  const char *path /* directory pathname */
- It Returns a DIR pointer or NULL on error

- # include<dirent.h>

  int closedir(DIR *dirp);

  DIR *dirp /*DIR pointer from opendir */
- It Returns 0 on success or -1 on error

readdir() System Call

- #include <dirent.h>
- struct dirent *readdir(DIR *dirp);
- /* DIR Pointer from opendir */
- It Returns structure or NULL on EOF or error

```
struct dirent
{
    ino_t d_ino;  /*  i-number */
    char d_name[];   /*  name */
};
```

```
int main(int argc,char *argv[])
{
    DIR *ds; struct dirent *dir;
    ds = opendir(argv[1]);
    if(ds == NULL)
       printf("directory %s is not
    opened\n",argv[1]);
    else
            printf("ds = %ld\n",ds);
    printf("list of files and directories\n");
    while((dir = readdir(ds)) != NULL)
            printf("%s\n",dir->d_name);
    if((cld = closedir(ds)) == 0)
            printf("%s is successfully
    closed\n",argv[1]);
    else
        printf("%s is not successfully
    closed\n",argv[1]);
}
```