USP Lab – Week 1

1. Execute the following commands in unix with different options and note down the output and comments.
2. ls
3. ps
4. cat
5. df
6. du
7. grep
8. find
9. gcc
10. Differentiate between internal and external commands of unix.
11. Write a C program to read from keyboard and display on monitor screen.
12. Write a C program to simulate copy command that copies contents of one file into another.
13. Write a C program to simulate a calculator reading the operation and operands from command line.

USP Lab – Week 2

1. Write a C program to read from standard input and display on standard output.

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<fcntl.h>

#define BUFFSIZE 100

int main()

{

int n;

char buf[BUFFSIZE];

while ((n = read(STDIN\_FILENO, buf, BUFFSIZE)) > 0)

if (write(STDOUT\_FILENO, buf, n) != n)

printf("write error");

if (n < 0)

printf("read error");

exit(0);

}

1. Write a program to read n characters from a file and append them back to the same file using dup2 function.

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int fd1=0,fd2=0;

char buf[50];

if((fd1=open("t12.txt",O\_RDWR,0))<0)

printf("file open error");

fd2=dup(fd1);

printf("%d %d \n",fd1, fd2);

read(fd1,buf,10);

lseek(fd2, 0L, SEEK\_END);

write(fd2, buf, 10);

printf("%s\n",outbuf);

return 0;

}

1. Write a program
   1. to read first 20 characters from a file
   2. seek to 10th byte from the beginning and display 20 characters from there
   3. seek 10 bytes ahead from the current file offset and display 20 characters
   4. display the file size

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

int main()

{

int file=0, n;

char buffer[25];

if((file=open("testfile.txt",O\_RDONLY)) < -1)

printf(“file open error\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if(lseek(file,10,SEEK\_SET) < 0)

printf(“lseek operation to beginning of file failed\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if(lseek(file,10,SEEK\_CUR) < 0)

printf(“lseek operation to beginning of file failed\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if((n = lseek(file,0,SEEK\_END)) <0)

printf(“lseek operation to end of file failed\n”);

printf("size of file is %d bytes\n",n);

close(file);

return 0;

}

1. Write a program to display the file content in reverse order using lseek system call.

#include<stdlib.h>

#include<stdio.h>

#include<fcntl.h>

#include<string.h>

#include<sys/stat.h>

#include<unistd.h>

int main(int argc, char \*argv[])

{

int source, dest, n;

char buf;

int filesize;

int i;

if (argc != 3) {

fprintf(stderr, "usage %s <source> <dest>", argv[0]);

exit(-1);

}

if ((source = open(argv[1], O\_RDONLY)) < 0)

{ fprintf(stderr, "can't open source\n");

exit(-1);

}

if ((dest = open(argv[2], O\_WRONLY | O\_CREAT | O\_TRUNC)) < 0)

{ fprintf(stderr, "can't create dest\n");

exit(-1);

}

filesize = lseek(source, (off\_t) 0, SEEK\_END);

printf("Source file size is %d\n", filesize);

for (i = filesize - 1; i >= 0; i--)

{

lseek(source, (off\_t) i, SEEK\_SET);

if ((n = read(source, &buf, 1)) != 1) {

fprintf(stderr, "can't read 1 byte");

exit(-1);

}

if ((n = write(dest, &buf, 1)) != 1) {

fprintf(stderr, "can't write 1 byte");

exit(-1);

}

}

write(STDOUT\_FILENO, "DONE\n", 5);

close(source);

close(dest);

return 0;

}

USP Lab – Week 3

1. Program to print file types

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

int main(int argc, char \*argv[])

{

int i;

struct stat buf;

char \*ptr;

for (i = 1; i < argc; i++)

{

printf("%s: ", argv[i]);

if (lstat(argv[i], &buf) < 0)

{

err\_ret("lstat error");

continue;

}

if (S\_ISREG(buf.st\_mode))

ptr = "regular";

else if (S\_ISDIR(buf.st\_mode))

ptr = "directory";

else if (S\_ISCHR(buf.st\_mode))

ptr = "character special";

else if (S\_ISBLK(buf.st\_mode))

ptr = "block special";

else if (S\_ISFIFO(buf.st\_mode))

ptr = "fifo";

else if (S\_ISLNK(buf.st\_mode))

ptr = "symbolic link";

else if (S\_ISSOCK(buf.st\_mode))

ptr = "socket";

else

ptr = "\*\* unknown mode \*\*";

printf("%s\n", ptr);

}

exit(0);

}

1. Program to print file access permissions.( Write a program to display various details of a file using stat structure(At least 5 fields)

#include <unistd.h>

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

int main(int argc, char \*\*argv)

{

if(argc != 2)

return 1;

struct stat fileStat;

if(stat(argv[1],&fileStat) < 0)

return 1;

printf("Information for %s\n",argv[1]);

printf("---------------------------\n");

printf("File Size: \t\t%d bytes\n",fileStat.st\_size);

printf("Number of Links: \t%d\n",fileStat.st\_nlink);

printf("File inode: \t\t%d\n",fileStat.st\_ino);

printf("File Permissions: \t");

printf( (S\_ISDIR(fileStat.st\_mode)) ? "d" : "-");

printf( (fileStat.st\_mode & S\_IRUSR) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWUSR) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXUSR) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IRGRP) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWGRP) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXGRP) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IROTH) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWOTH) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXOTH) ? "x" : "-");

printf("\n\n");

printf("The file %s a symbolic link\n", (S\_ISLNK(fileStat.st\_mode)) ? "is" : "is not");

return 0;

}

1. Programs to demonstrate usage of access, umask and chmod functions.

#include <fcntl.h>

int main(int argc, char \*argv[])

{

if (argc != 2)

printf("usage: argv[0] <pathname>");

if (access(argv[1], R\_OK) < 0)

printf ("access error for %s", argv[1]);

else

printf("read access OK\n");

if (open(argv[1], O\_RDONLY) < 0)

printf ("open error for %s", argv[1]);

else

printf("open for reading OK\n");

exit(0);

}

**Example of umask function**

#include <fcntl.h>

#define RWRWRW (S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IWGRP|S\_IROTH|S\_IWOTH)

int main()

{

umask(0);

if (creat("foo", RWRWRW) < 0)

printf("creat error for foo");

umask(S\_IRGRP | S\_IWGRP | S\_IROTH | S\_IWOTH);

if (creat("bar", RWRWRW) < 0)

printf("creat error for bar");

exit(0);

}

**Example of chmod function**

int main()

{

struct stat statbuf;

/\* turn on set-group-ID and turn off group-execute \*/

if (stat("foo", &statbuf) < 0)

err\_sys("stat error for foo");

if (chmod("foo", (statbuf.st\_mode & ~S\_IXGRP) | S\_ISGID) < 0)

err\_sys("chmod error for foo");

/\* set absolute mode to "rw-r--r--" \*/

if (chmod("bar", S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH) < 0)

err\_sys("chmod error for bar");

exit(0);

}

USP Lab – Week 4

1. Program to demonstrate the creation of hard links and the various properties of hard links

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

int main(int argc, char\*argv[])

{

        if(argc==3)

        {

        printf("Hard linking %s and %s",argv[1],argv[2]);

        if(link(argv[1],argv[2])==0)

                printf("\nHard link created");

        else

                printf("\nLink not created");

        }

        else if(argc==4)

        {

        printf("Soft linking %s and %s",argv[1],argv[2]);

        if(symlink(argv[1],argv[2])==0)

                printf("\nSoft link created");

        else

                printf("\nLink not created");

        }

}

1. Program to demonstrate the creation of soft links and the various properties of hard links
2. Write a program to implement ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.

|  |
| --- |
| #include <stdio.h> |
|  |

|  |
| --- |
| #include <unistd.h> |
|  |

|  |
| --- |
| #include <fcntl.h> |
|  |

|  |
| --- |
| #include <dirent.h> |
|  |

|  |
| --- |
| #include <time.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main(int argc,char\* argv[]) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| struct dirent \*dir; |
|  |

|  |
| --- |
| struct stat mystat; |
|  |

|  |
| --- |
| DIR \*dp; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| dp = opendir("."); |
|  |

|  |
| --- |
| if(dp) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| while(dir = readdir(dp)) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| stat(dir->d\_name,&mystat); |
|  |

|  |
| --- |
| // inode mode uid guid access\_time |
|  |

|  |
| --- |
| printf("%ld %o %d %d %s %s\n", |
|  |

|  |
| --- |
| mystat.st\_ino,mystat.st\_mode,mystat.st\_uid,mystat.st\_gid,ctime(&mystat.st\_atime),dir->d\_name); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

}

1. Write a program to remove empty files from the given directory.

|  |
| --- |
| #include <stdio.h> |
|  |

|  |
| --- |
| #include <fcntl.h> |
|  |

|  |
| --- |
| #include <unistd.h> |
|  |

|  |
| --- |
| #include <dirent.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| DIR \*dp; |
|  |

|  |
| --- |
| struct dirent \*dir; |
|  |

|  |
| --- |
| int fd,n; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| dp = opendir("."); //open current directory |
|  |

|  |
| --- |
| if(dp) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| while((dir = readdir(dp)) != NULL) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| fd = open(dir->d\_name,O\_RDWR,0777); |
|  |

|  |
| --- |
| n = lseek(fd,0,SEEK\_END); |
|  |

|  |
| --- |
| if(!n) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| unlink(dir->d\_name); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

}

1. Write a program to Copy access and modification time of a file to another file using utime function.

#include "apue.h"

#include <fcntl.h>

#include <utime.h>

int main(int argc, char \*argv[])

{

int i, fd;

struct stat statbuf;

struct utimbuf timebuf;

for (i = 1; i < argc; i++) {

if (stat(argv[i], &statbuf) < 0) { /\* fetch current times \*/

printf("%s: stat error", argv[i]);

continue;

}

if ((fd = open(argv[i], O\_RDWR | O\_TRUNC)) < 0) { /\* truncate \*/

printf ("%s: open error", argv[i]);

continue;

}

close(fd);

timebuf.actime = statbuf.st\_atime;

timebuf.modtime = statbuf.st\_mtime;

if (utime(argv[i], &timebuf) < 0)

{ /\* reset times \*/

printf("%s: utime error", argv[i]);

continue;

}

}

exit(0);

}

# OR

|  |
| --- |
| #include <stdio.h> |
|  |

|  |
| --- |
| #include <sys/stat.h> |
|  |

|  |
| --- |
| #include <sys/types.h> |
|  |

|  |
| --- |
| #include <unistd.h> |
|  |

|  |
| --- |
| #include <utime.h> |
|  |

|  |
| --- |
| #include <time.h> |
|  |

|  |
| --- |
| #include <fcntl.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main(int argc,char\* argv[]) //copying ctime and mtime of argv[2] to argv[1] |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| int fd; |
|  |

|  |
| --- |
| struct stat statbuf\_1; |
|  |

|  |
| --- |
| struct stat statbuf\_2; |
|  |

|  |
| --- |
| struct utimbuf times; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(stat(argv[1],&statbuf\_1)<0) |
|  |

|  |
| --- |
| printf("Error!\n"); |
|  |

|  |
| --- |
| if(stat(argv[2],&statbuf\_2)<0) |
|  |

|  |
| --- |
| printf("Error!\n"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| printf("Before Copying ...\n"); |
|  |

|  |
| --- |
| printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| times.modtime = statbuf\_2.st\_mtime; |
|  |

|  |
| --- |
| times.actime = statbuf\_2.st\_mtime; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(utime(argv[1],&times)<0) |
|  |

|  |
| --- |
| printf("Error copying time \n"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(stat(argv[1],&statbuf\_1)<0) |
|  |

|  |
| --- |
| printf("Error!\n"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| printf("After Copying ...\n"); |
|  |

|  |
| --- |
| printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime)); |
|  |

|  |
| --- |
|  |
|  |

}

Week 6

1. C program to simulate copy command by accepting the filenames from command line. Report all errors.

#include<stdio.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc, char \*argv[])

{char buf[100];

int fd1,fd2;

off\_t size,ret,set;

ssize\_t readdata,writedata;

if(argc<3)

printf("TOO FEW ARGUMENTS");

fd1=open(argv[1],O\_RDONLY); //Open file 1

if(fd1==-1)

printf("ERROR IN OPENING FILE: FILE DOES NOT EXIST \n");

else

printf("FILE 1 OPENED SUCCESSFULLY \n");

fd2=open(argv[2],O\_WRONLY | O\_CREAT | O\_TRUNC, 0666); //open file 2 in read-write mode, truncate its length to 0, create the file if it does not exist, 0666 is the access permission for the created file. order is important.

if(fd2==-1)

printf("ERROR IN OPENING FILE");

else

printf("FILE 2 OPENED SUCCESSFULLY \n");

size=lseek(fd1,0L,SEEK\_END); //obtain the size of file 1 using lseek

if(size==-1)

printf("ERROR: COULD NOT OBTAIN FILE SIZE \n");

else

printf("FILE SIZE OF FILE 1 OBTAINED \n");

ret=lseek(fd1,0L,SEEK\_SET); //change the current pointer to the beginning of the file

if(ret==-1)

printf("RETRACE FAILED \n");

readdata=read(fd1,buf,size); //read data equal to the size of the first file

if(readdata==-1)

printf("ERROR IN READING FILE CONTENTS \n");

writedata=write(fd2,buf,size); //write the data to file 2 from buffer after read

if(writedata!=size)

printf("ERROR IN COPYING FILE");

else

printf("FILE COPIED SUCCESSFULLY");

return 0;

}

2. C program to display environment variables using global variable environ.

#include<stdio.h>

#include<stdlib.h>

//extern char \*\*environ;

int main() {

int i = 1;

printf("test\n");

const char\* s = getenv("PATH");

const char\* p = getenv("PWD");

const char\* l = getenv("LOGNAME");

printf("PATH :%s\n",(s!=NULL)? s : "getenv returned NULL");

printf("PWD :%s\n",(p!=NULL)? p : "getenv returned NULL");

printf("LOGNAME :%s\n",(l!=NULL)? l : "getenv returned NULL");

printf("end test\n");

return 0;

}

3. C program to display environment variables using APIs in C language.

4. C program to access the shell environment variable(SHELL). Print its value. Set it to /bin/csh and print the new value to standard output.

5. C program to illustrate effect of setjmp and longjmp functions on register, volatile and automatic variables.

#include <setjmp.h>

#include<stdio.h>

#include<stdlib.h>

static void f1(int, int, int, int);

static void f2(void);

static jmp\_buf jmpbuffer;

static int globval;

int main(void)

{

int autoval;

register int regival;

volatile int volaval;

static int statval;

globval = 1; autoval = 2; regival = 3; volaval = 4; statval = 5;

if (setjmp(jmpbuffer) != 0)

{

printf("after longjmp:\n");

printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, autoval, regival, volaval, statval);

exit(0);

}

/\*

\* Change variables after setjmp, but before longjmp.

\*/

globval = 95; autoval = 96; regival = 97; volaval = 98;

statval = 99;

f1(autoval, regival, volaval, statval); /\* never returns \*/

exit(0);

}

static void f1(int i, int j, int k, int l)

{

printf("in f1():\n");

printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, i, j, k, l);

globval=10000;

j=10000;

f2();

}

static void f2(void)

{

longjmp(jmpbuffer, 1);

}

6. C program to create a new process and demonstrate the working of fork function.

Week 8 Assignment

1. C program to create a new process and demonstrate the working of fork function.

2. C program to create a new process and demonstrate the working of vfork function.

3. C program to demonstrate the working of wait function.

#include "apue.h"

#include <sys/wait.h>

int main(void)

{

pid\_t pid;

int status;

if ((pid = fork()) < 0)

err\_sys("fork error");

else if (pid == 0) /\* child \*/

exit(7);

if (wait(&status) != pid) /\* wait for child \*/

err\_sys("wait error");

pr\_exit(status); /\* and print its status \*/

if ((pid = fork()) < 0)

err\_sys("fork error");

else if (pid == 0) /\* child \*/

abort(); /\* generates SIGABRT \*/

if (wait(&status) != pid) /\* wait for child \*/

err\_sys("wait error");

pr\_exit(status); /\* and print its status \*/

if ((pid = fork()) < 0)

err\_sys("fork error");

else if (pid == 0) /\* child \*/

status /= 0; /\* divide by 0 generates SIGFPE \*/

if (wait(&status) != pid) /\* wait for child \*/

err\_sys("wait error");

pr\_exit(status); /\* and print its status \*/

exit(0);

}

4. C program to avoid zombie status of a process.

#include<stdio.h>

#include<stdlib.h>

#include <sys/wait.h>

int

main(void)

{

pid\_t pid;

if ((pid = fork()) < 0) {

err\_sys("fork error");

}

else if (pid == 0) { /\* first child \*/

if ((pid = fork()) < 0)

err\_sys("fork error");

else if (pid > 0)

exit(0);

sleep(2);

printf("second child, parent pid = %ld\n", (long)getppid());

exit(0);

}

if (waitpid(pid, NULL, 0) != pid)

err\_sys("waitpid error");

exit(0);

}

5. C program to demonstrate race condition among parent and child processes.

6. C program to create a new process and demonstrate the working of exec function.

#include<stdio.h>

#include<stdlib.h>

//extern char \*\*environ;

int main() {

int i = 1;

printf("test\n");

const char\* s = getenv("PATH");

const char\* p = getenv("PWD");

const char\* l = getenv("LOGNAME");

printf("PATH :%s\n",(s!=NULL)? s : "getenv returned NULL");

printf("PWD :%s\n",(p!=NULL)? p : "getenv returned NULL");

printf("LOGNAME :%s\n",(l!=NULL)? l : "getenv returned NULL");

printf("end test\n");

return 0;

}

Week 10

/\*Write a program (use signal system call)

i. which calls a signal handler on SIGINT signal and then reset the default action of the

SIGINT signal

ii. Which ignores SIGINT signal and then reset the default action of SIGINT signal\*/

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

void callback()

{

printf("Intrrupt Received !\n");

(void)signal(SIGINT,SIG\_DFL);

}

int main()

{

int ch,i=0;

printf("Enter choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1 : (void)signal(SIGINT,callback);

break;

case 2 : (void)signal(SIGINT,SIG\_IGN);

break;

}

while(1)

{

sleep(1);

printf("Press CTRL+C ...\n");

i++;

if(i == 10 && ch == 2)

(void) signal(SIGINT,SIG\_DFL);

}

return 0;

}

Write a program to create a coprocess

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<fcntl.h>

#define BUFFSIZE 100

int main()

{

int n;

char buf[BUFFSIZE];

while ((n = read(STDIN\_FILENO, buf, BUFFSIZE)) > 0)

if (write(STDOUT\_FILENO, buf, n) != n)

printf("write error");

if (n < 0)

printf("read error");

exit(0);

}

/\*Write a program using sigaction system call which calls a signal handler on SIGINT signal

and then reset the default action of the SIGINT signal\*/

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

struct sigaction sig;

void handler(int val)

{

printf("Interrupt Received!\n");

sig.sa\_handler = SIG\_DFL;

sigaction(SIGINT,&sig,0);

}

int main()

{

sig.sa\_flags = 0;

sigemptyset(&sig.sa\_mask);

sigaddset(&sig.sa\_mask,SIGINT); // listen only for SIGNIT

sig.sa\_handler = handler;

sigaction(SIGINT,&sig,0);

while(1)

{

printf("Progress is Happiness!\n");

sleep(1);

}

}

Program to establish client server communication using pipes

Client program

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <string.h>

#include <fcntl.h>

#define MAX 80

int main(int argc,char\* argv[])

{

char buffer[MAX];

int in,out;

int n;

in = open("server\_to\_client",O\_RDWR,0777);

out = open("client\_to\_server",O\_RDWR,0777);

printf("Sending Message\n");

n = strlen(argv[1]);

write(out,argv[1],n);

read(in,buffer,n);

buffer[n]='\0';

printf("Message Received : %s\n",buffer);

close(in);

close(out);

}

server program

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

#include <string.h>

#define MAX 80

int main()

{ char buffer[MAX];

int in,out,n;

mkfifo("server\_to\_client",0777);

mkfifo("client\_to\_server",0777);

while(1)

{ in = open("client\_to\_server",O\_RDWR,0777);

out = open("server\_to\_client",O\_RDWR,0777);

memset(buffer,0,MAX);

printf("Waiting for Message\n");

n = read(in,buffer,MAX);

printf("Message : %s\n",buffer);

buffer[0] = toupper(buffer[0]);

printf("Sending Reply\n");

write(out,buffer,n);

close(in);

close(out);

}

return 0;

}