WEEK 5

1. Write a C program to display environment variables using global variable environ.

#include <stdio.h>

#include <stdlib.h>

extern char \*\*environ;

int main()

{

int i = 0;

printf("Listing all environment variables:\n\n");

while (environ[i] != NULL)

{

printf("%s\n", environ[i]);

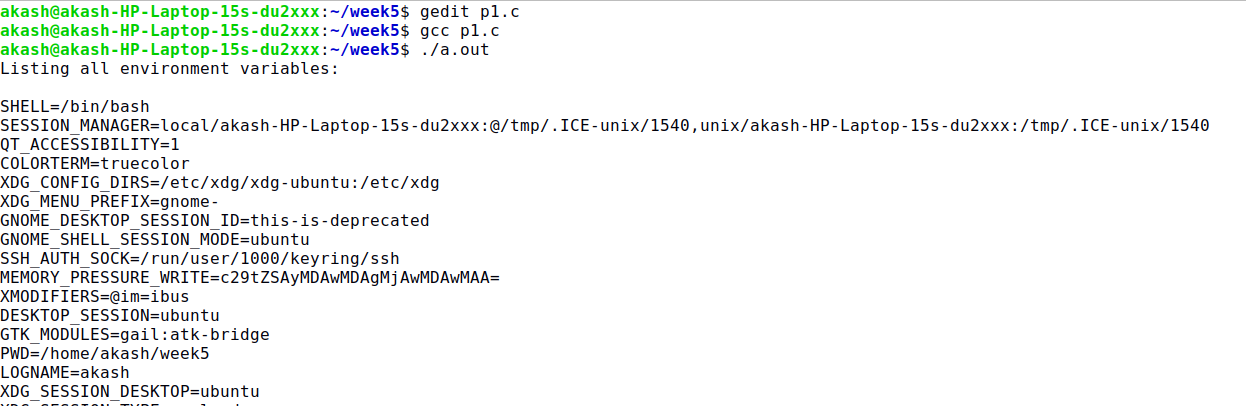
i++;

}

printf("\nEnd of environment variables list.\n");

return 0;

}



#include<stdio.h>

#include<stdlib.h>

int main()

{

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");

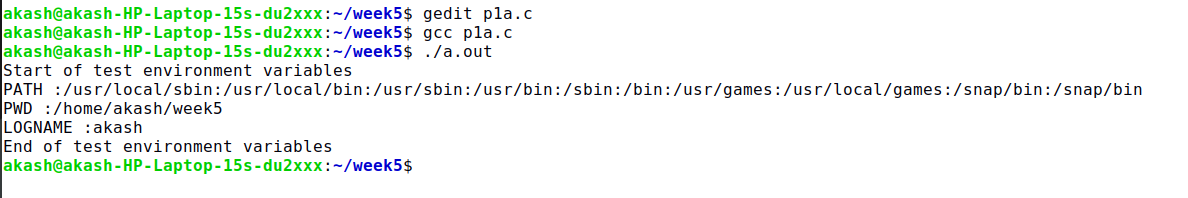
printf("PWD :%s\n",(p!=NULL)? p : "getenv returned NULL");

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

printf("end test\n");

return 0;

}



1. Write a C program to illustrate the effect of setjmp and longjmp functions on register and volatile 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();

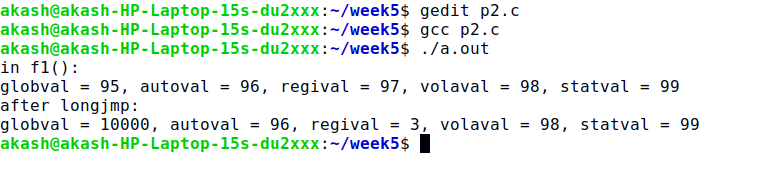
}

static void f2(void)

{

longjmp(jmpbuffer, 1);

}



1. Write a 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[5000];

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)

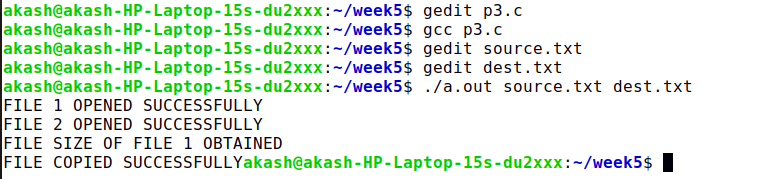
printf("ERROR IN COPYING FILE");

else

printf("FILE COPIED SUCCESSFULLY");

return 0;

}



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

/\*showing how changes to variables in a child process do not affect the value of the variables in the parent process.\*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/stat.h>

int glob = 6; /\* external variable in initialized data \*/

char buf[]="a write to stdout\n";

int main(void)

{

int var; /\* automatic variable on the stack \*/

pid\_t pid;

var = 88;

if(write(STDOUT\_FILENO,buf,sizeof(buf)-1)!=sizeof(buf)-1)

printf("write error");

printf("before fork\n"); /\* we don't flush stdio \*/

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

{

printf("fork error");

}

else

if (pid == 0)

{

/\* child \*/

glob++; /\* modify parent's variables \*/

var++;

}

else

{

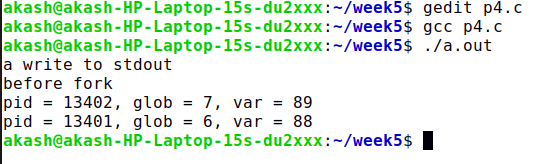
sleep(2);

}

printf("pid = %d, glob = %d, var = %d\n", getpid(), glob, var);

exit(0);

}



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

#include<unistd.h>

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<stdlib.h>

int glob = 6; /\* external variable in initialized data \*/

int main(void)

{

int var; /\* automatic variable on the stack \*/

pid\_t pid;

var = 88;

printf("before vfork\n"); /\* we don't flush stdio \*/

if ((pid = vfork()) < 0)

{

printf("vfork error");

}

else

if (pid == 0)

{

/\* child \*/

glob++; /\* modify parent's variables \*/

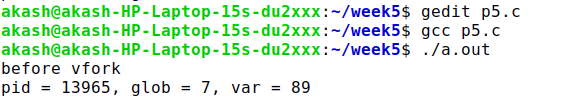
var++;

\_exit(0); /\* child terminates \*/

} /\* \* Parent continues here. \*/

printf("pid = %d, glob = %d, var = %d\n", getpid(), glob, var); exit(0);

}



6. Write a C Program to demonstrate various exit statuses and wait function

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <errno.h>

#include <signal.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);