**SECURE CODING LAB 4**

**Name:Ramya Ajay**

**Roll No:CB.EN.P2CYS22004**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM 1 – SYSCALLS**

#include<stdio.h> #include<unistd.h> #include<stdlib.h>

int main()

{

int pid,pid1,pid2; pid=fork(); if(pid==1)

{

printf("ERROR IN PROCESS CREATION \n");

exit(1);

}

if(pid!=0)

{

pid1=getpid();

printf("\n the parent process ID is %d\n", pid1);

}

else

{

pid2=getpid();

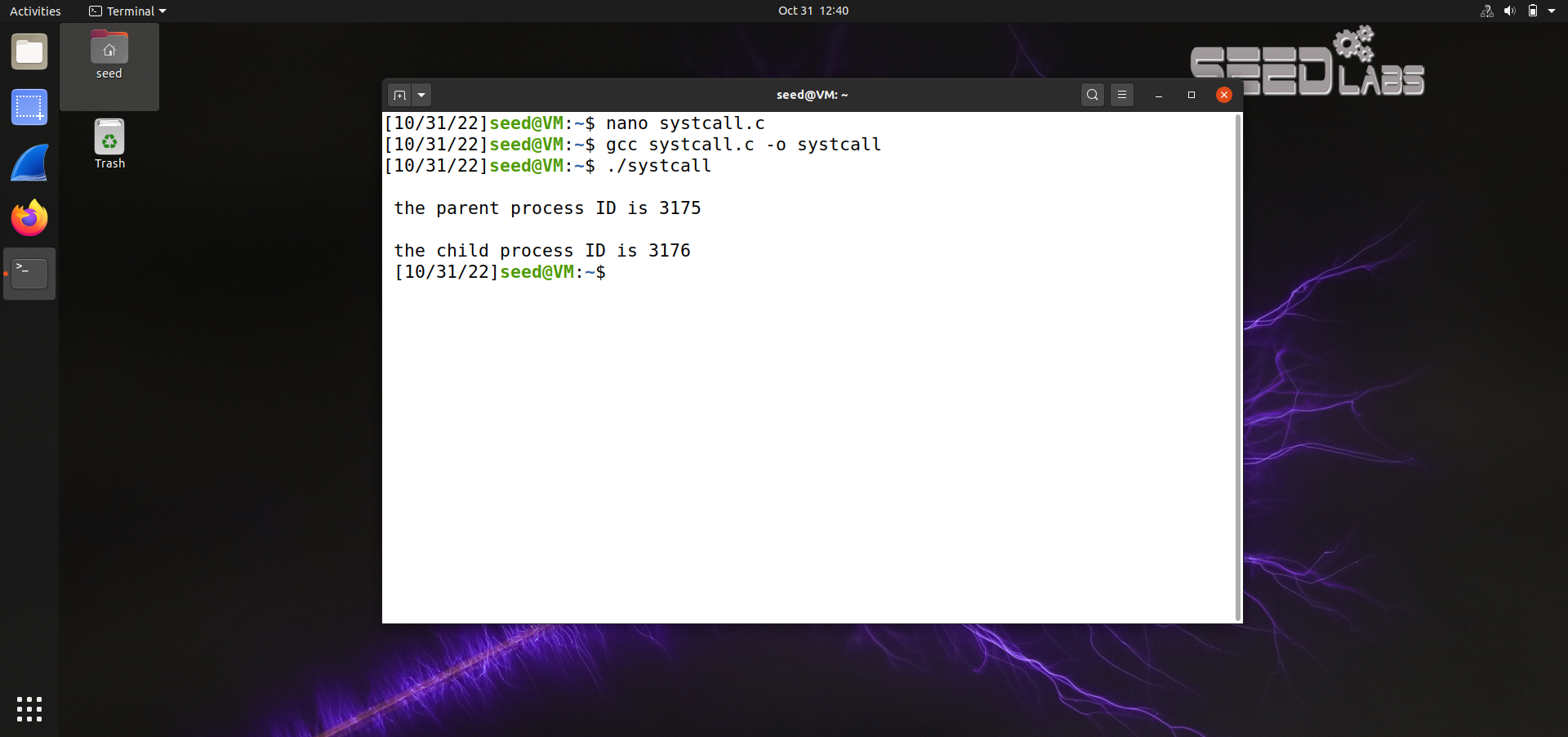
printf("\n the child process ID is %d\n ", pid2);

}

return 0;

}

**OUTPUT -**



1. Write your own version of printf named myprintfunction().

A) It should be able to accept various types of parameters such as char, int, double, etc.

B) Bonus : The function should be able to accept different parameter count. The first parameter says the count of parameters, followed by actual parameters

#include <stdio.h>

#include <stdlib.h>

#include <stdarg.h>

#include <string.h>

char \*\_strrev (char \*str)

{

int i;

int len = 0; char c;

if (!str) return NULL;

while(str[len] != '\0'){ len++;

}

for(i = 0; i < (len/2); i++)

{

c = str[i];

str [i] = str[len - i - 1];

str[len - i - 1] = c;

}

return str;

}

char \* \_itoa(int i, char \*strout, int base)

{

char \*str = strout; int digit, sign = 0; if (i < 0)

{

sign = 1;

i \*= -1;

}

while(i)

{

digit = i % base;

\*str = (digit > 9) ? ('A'+ digit - 10) : '0' + digit; i = i / base; str ++;

}

if(sign)

{

\*str++ = '-';

}

\*str = '\0';

\_strrev(strout); return strout;

}

void myprintfunction(int n\_args, ...)

{

va\_list vl; int i = 0, j=0;

char buff[100]={0}, tmp[20]; char \* str\_arg;

va\_start(vl, n\_args);

const char \*str = va\_arg(vl, const char\*);

printf("Total count of parameters given : %d\n", n\_args);

while (str && str[i])

{

if(str[i] == '%')

{ i++;

switch (str[i])

{

case 'c':

{

buff[j] = (char)va\_arg( vl, int ); j++;

break;

}

case 'd':

{

\_itoa(va\_arg( vl, int ), tmp, 10); strcpy(&buff[j], tmp);

j += strlen(tmp); break;

}

case 'x':

{

\_itoa(va\_arg( vl, int ), tmp, 16); strcpy(&buff[j], tmp);

j += strlen(tmp); break;

}

case 'o':

{

\_itoa(va\_arg( vl, int ), tmp, 8); strcpy(&buff[j], tmp);

j += strlen(tmp); break;

}

case 's':

{

str\_arg = va\_arg( vl, char\* ); strcpy(&buff[j], str\_arg);

j += strlen(str\_arg); break;

}

}

}

else

{

buff[j] =str[i]; j++;

} i++;

}

fwrite(buff, j, 1, stdout); va\_end(vl);

}

int main()

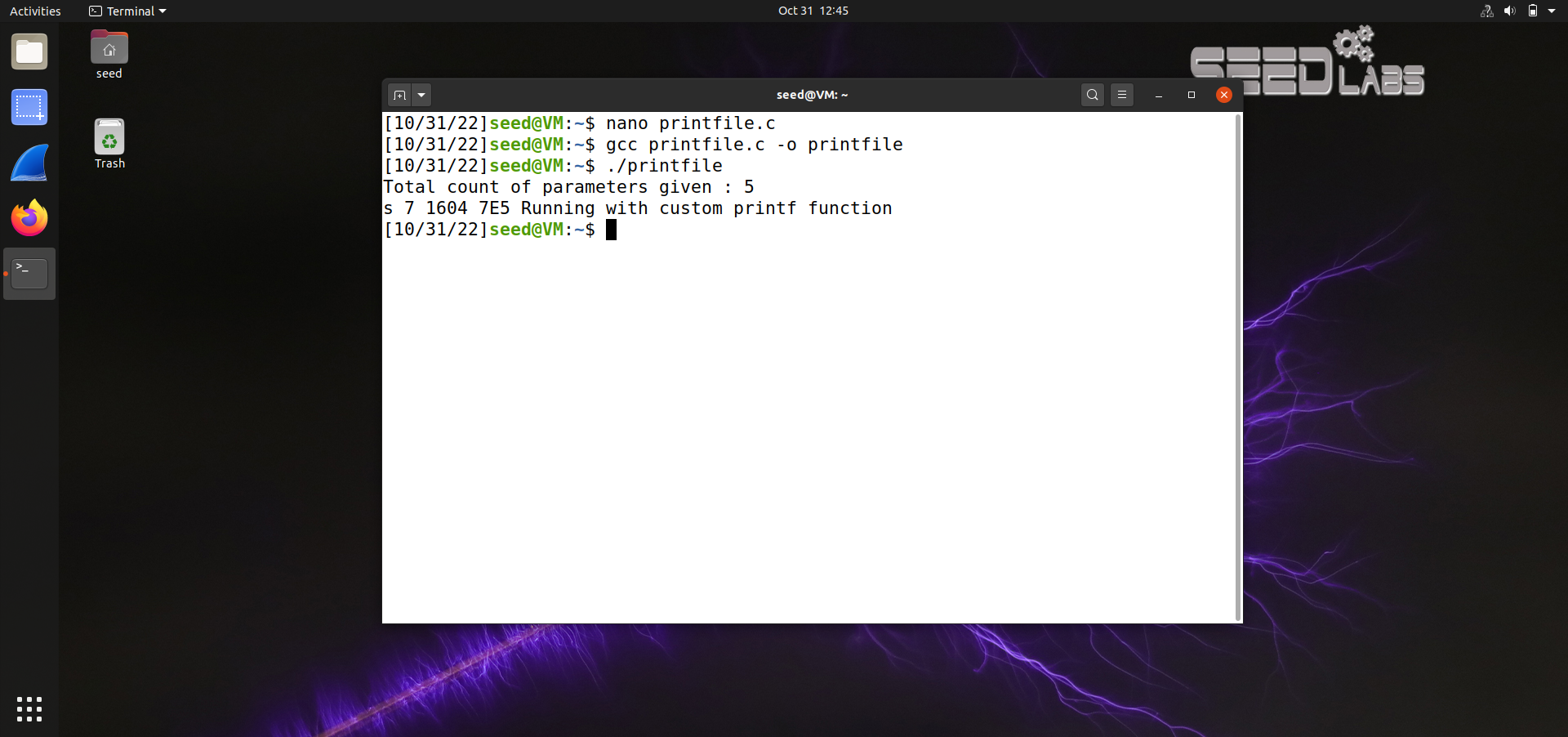
{

int n\_args=5;

myprintfunction(n\_args, "%c %d %o %x %s\n",'s', 007, 900, 2021,"Running with custom printf function");

return 0;

**OUTPUT:**



2. Write a program to read all txt files (that is files that ends with .txt) in the current directory and merge them all to one text file and return a file descriptor for the new file.

**PROGRAM -**

#include <dirent.h> #include <stdio.h> #include <string.h> int main(void)

{

DIR \*cdir;

char \*ptr1,\*ptr2, filelist[100][100], temp[100], c; int retn, k=0;

char filename[100] = ("mergefile.txt"); struct dirent \*dir;

cdir = opendir(".");

printf("Text files in the directory:- \n"); if (cdir)

{

while ((dir = readdir(cdir)) != NULL)

{

ptr1=strtok(dir->d\_name,"."); ptr2=strtok(NULL,"."); if(ptr2!=NULL)

{

retn=strcmp(ptr2,"txt"); if(retn==0)

{

strcpy(temp, ptr1); strcat(temp, ".txt"); strcpy(filelist[k++], temp); printf("%s\n", temp);

}

}

}

closedir(cdir);

}

FILE \*fptr1 = fopen(filename,"a"), \*fptr2[k]; for(int i=k-1; i>=0; i--)

{

FILE \*fptr2 = fopen(filelist[i], "r"); while((c = fgetc(fptr2)) != EOF) fputc(c, fptr1);

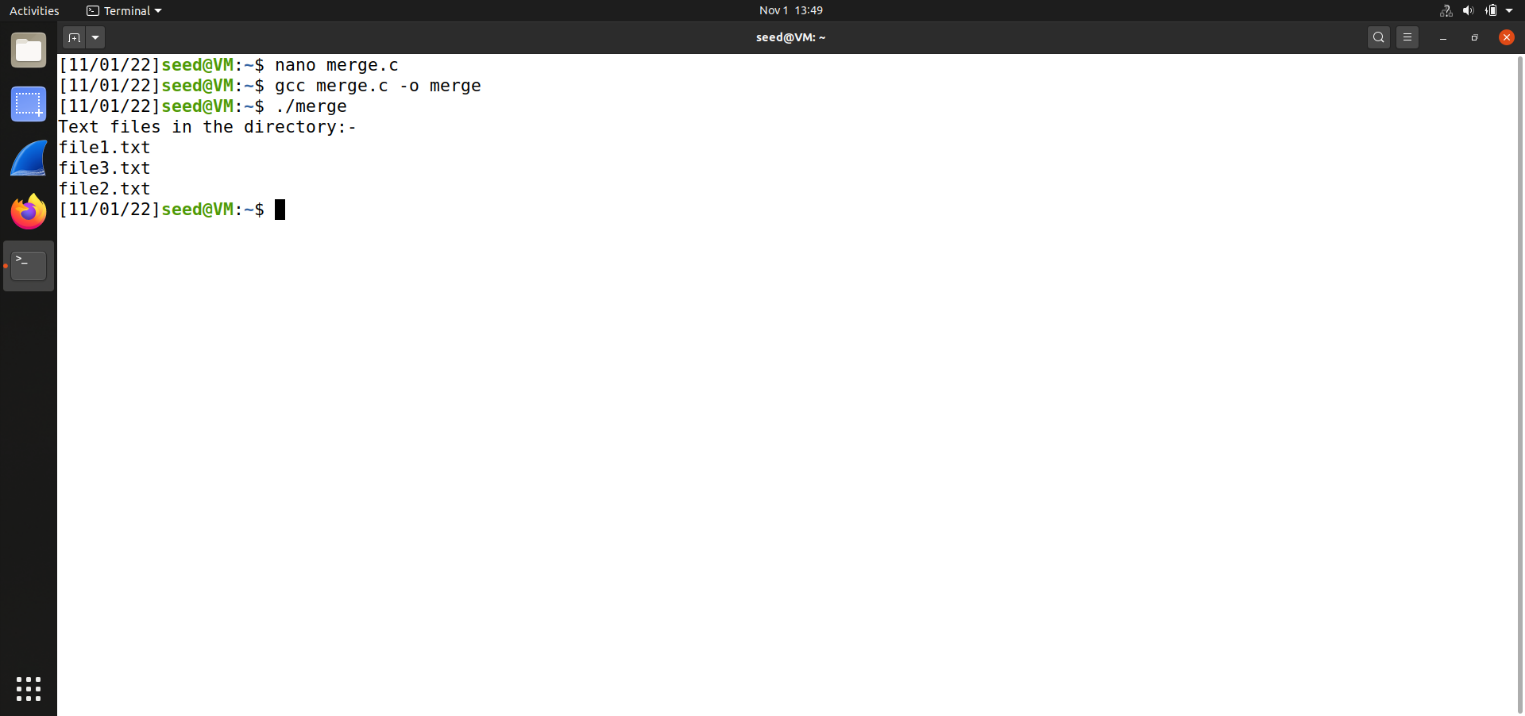
fclose(fptr2);

}

return 0;

}

**OUTPUT :**

****

**3. Write a program that will categorize all files in the current folder based on their file type. That is all .txt files in one folder called txt, all .bmp files in another folder called bmp etc. The argument to the program is a folder name.**

#include<dirent.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

int main(void)

{

DIR \*cdir;

char \*ptr1,\*ptr2, ext[100][100], c; char filename[50], filepath[100]; for(int i=0; i<100; i++) strcpy(ext[i], "0");

int retn;

struct dirent \*dir; cdir = opendir("."); if (cdir)

{

while ((dir = readdir(cdir)) != NULL)

{

ptr1=strtok(dir->d\_name,"."); ptr2=strtok(NULL,"."); if(ptr2!=NULL)

{

if(strcmp(ext[ptr2[0]-97], "0") == 0)

strcmp(ext[ptr2[0]-97], ptr2);

strcpy(filename, ptr1); strcat(filename, "."); strcat(filename, ptr2); mkdir(ptr2, 0755); strcpy(filepath, ptr2); strcat(filepath, "/"); strcat(filepath,

filename);

FILE \*fp1 = fopen(filepath, "w");

FILE \*fp2 = fopen(filename, "r");

while((c = fgetc(fp2)) != EOF) fputc(c, fp1);

}

}

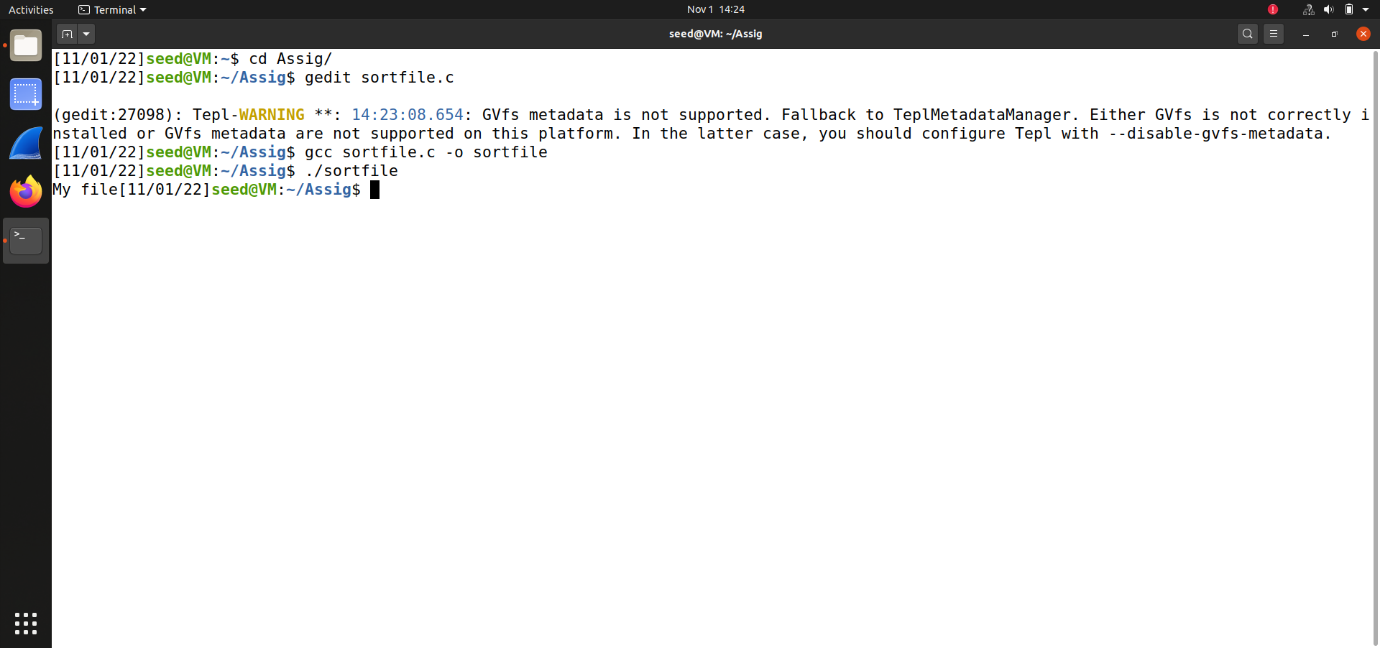
closedir(cdir);

}

return 0;

}

**OUTPUT:**

****

4.