Program – 1

Object - Write a program to display a file page wise assuming a page has 10 lines and each line has 80 characters.

Code :

```
#include <stdio.h>
#define MAX_LINE_LEN 80       // Define a constant for maximum line length
#define MAX_LINES_PER_PAGE 10 // Define a constant for maximum number of lines per page

int main(int argc, char *argv[])
{
    if (argc != 2)
    { // Check if the user has provided a filename argument
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    FILE *fp = fopen(argv[1], "r"); // Open the file for reading
    if (fp == NULL)
    { // Check if the file was opened successfully
        fprintf(stderr, "Error: could not open file %s\n", argv[1]);
        return 1;
    }
    char line[MAX_LINE_LEN]; // Define a buffer for reading each line of the file
    int line_count = 0;      // Keep track of the number of lines printed so far
    while (fgets(line, MAX_LINE_LEN, fp) != NULL)
    {                        // Read each line of the file
        printf("%s\n", line); // Print the line to the console
        line_count++;         // Increment the line count
        if (line_count >= MAX_LINES_PER_PAGE)
        { // Check if we've printed enough lines for the current page
            printf("\nPress ENTER to continue...\n");
            getchar();      // Wait for the user to press ENTER
            line_count = 0; // Reset the line count for the next page
        }
    }
    fclose(fp); // Close the file
    return 0;   // Exit the program
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_1$ ./Cprog1 sample.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut hendrerit nibh et l
acinia tincidunt.

Cras condimentum, tortor vel vestibulum tincidunt, quam ipsum dignissim massa,
at porta dolor erat nec metus.

Donec eu sapien placerat, malesuada mauris a, vestibulum nisl.

Phasellus ex leo, semper eu mauris ut, varius laoreet est.

Aliquam viverra scelerisque odio, ac tristique risus. Nullam vitae tincidunt ar
cu, non gravida lacus.

Vivamus vestibulum elit fermentum libero laoreet bibendum.

Aliquam nec felis quis velit pellentesque auctor. Praesent porttitor pretium ip

Press ENTER to continue...

sum sit amet venenatis.

Donec et elit non nisl ultricies sollicitudin. Donec ac ex ac arcu laoreet pulv
inar porta ac.
```

Program – 2

Object - Write a Program which converts all the small case letters in a file into appropriate capital letters.

Code :

```
#include <stdio.h>
int main(int argc, char *argv[])
{
   // Check if the program is run with the correct number of arguments
   if (argc != 2)
   {
      // Print an error message to standard error output (stderr)
      fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
      // Return an error code (-1) to indicate that the program did not run successfully
      return -1;
   }
   // Open the file given as the argument in read/write mode
   FILE *fp = fopen(argv[1], "r+");
   // Check if the file was successfully opened
   if (fp == NULL)
   {
      // Print an error message, including a description of the error that occurred
      perror("Error opening file");
      // Return an error code (-1) to indicate that the program did not run successfully
      return -1;
   }
   int c;
   // Read each character in the file until the end of the file is reached (EOF)
   while ((c = fgetc(fp)) != EOF)
   {
      // Check if the character is a lowercase letter
      if (c >= 'a' && c <= 'z')
      {
         // Move the file pointer back one position to overwrite the lowercase letter
         fseek(fp, -1, SEEK_CUR);
         // Write the uppercase version of the letter to the file
         fputc(c - 'a' + 'A', fp);
      }
   }
   fclose(fp);
```

```
    return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_2$ cat sample.txt
This is sample text
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_2$ ./Cprog2Second sample.txt
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_2$ cat sample.txt
THIS IS SAMPLE TEXT
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_2$
```

Program – 3

Object - Write a program to print the details of the system (use uname sys call).

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/utsname.h>
// included header file `stdlib.h` for exit failure and sucess macros value, `sys/utsname.h` to get system information
int main()
{
    struct utsname sys_data;
    if (uname(&sys_data) != 0) // checks if there is any error in calling uname i.e. getting system info as uname return a dtructured info of the system
    {
        fprintf(stderr, "Error calling uname\n");
        return EXIT_FAILURE; // EXIT_FAILURE is a macro defined in stdlib.h
    }
    printf("Operating System : %s\n", sys_data.sysname);
    printf("Nodename : %s\n", sys_data.nodename);
    printf("Release : %s\n", sys_data.release);
    printf("Version : %s\n", sys_data.version);
    printf("Machine : %s\n", sys_data.machine);
    return EXIT_SUCCESS; // EXIT_SUCESS is a macro defined in stdlib.h
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_3$ ./prog3
Operating System : Linux
Nodename : AP
Release : 5.19.0-41-generic
Version : #42~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Apr 18 17:40:00 UTC 2
Machine : x86_64
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_3$
```

Program – 4

Object -Write a program which will print the list of environment variable and also print the value of the PATH system variable.

Code :

```
#include <stdio.h>  //Include standard input/output library
#include <stdlib.h> //Include standard library

extern char **environ; // Declare an external variable environ of type char**

int main()
{ // Declare the main function

   char *path; // Declare a pointer variable named path of type char*
   int i;      // Declare an integer variable named i

   printf("The environment list is as follows: \n"); // Print a message on the screen

   for (i = 0; environ[i] != NULL; i++) // For loop to traverse all the environment
variables
      printf("Environ[%d] : %s\n", i, environ[i]); // Print the environment variables

   path = getenv("PATH"); // Retrieve value of the PATH environment variable and store it

   if ((path == NULL))      // Check if path is null or not
      printf("Environment variable not set\n");  // Print a message on the screen if path is
null
   else   // If path is not null
         // Print the value of the PATH environment variable on the screen
      printf("The value of Path variable : %s", path);

   return 0; // Return 0 as the program executed successfully
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_4$ ./prog4
The environment list is as follows:
Environ[0] : SHELL=/bin/bash
Environ[1] : SESSION_MANAGER=local/AP:@/tmp/.ICE-unix/1639,unix/AP:/tmp/.ICE-unix/1639
Environ[2] : QT_ACCESSIBILITY=1
Environ[3] : COLORTERM=truecolor
Environ[4] : XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
Environ[5] : SSH_AGENT_LAUNCHER=gnome-keyring
Environ[6] : XDG_MENU_PREFIX=gnome-
Environ[7] : TERM_PROGRAM_VERSION=1.77.3
Environ[8] : GNOME_DESKTOP_SESSION_ID=this-is-deprecated
Environ[9] : LANGUAGE=en_IN:en
Environ[10] : GNOME_SHELL_SESSION_MODE=ubuntu
Environ[11] : SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
Environ[12] : XMODIFIERS=@im=ibus
Environ[13] : DESKTOP_SESSION=ubuntu
Environ[14] : BAMF_DESKTOP_FILE_HINT=/var/lib/snapd/desktop/applications/code_code.desktop
Environ[15] : GTK_MODULES=gail:atk-bridge
Environ[16] : PWD=/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_4
Environ[17] : GSETTINGS_SCHEMA_DIR=/home/ap-73/snap/code/126/.local/share/glib-2.0/schemas
Environ[18] : XDG_SESSION_DESKTOP=ubuntu
Environ[19] : LOGNAME=ap-73
Environ[20] : GTK_EXE_PREFIX=/snap/code/126/usr
Environ[21] : XDG_SESSION_TYPE=wayland
Environ[22] : SYSTEMD_EXEC_PID=1670
Environ[23] : XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.W2CT31
Environ[24] : VSCODE_GIT_ASKPASS_NODE=/snap/code/126/usr/share/code/code
Environ[25] : IM_CONFIG_CHECK_ENV=1
Environ[26] : GJS_DEBUG_TOPICS=JS ERROR;JS LOG
Environ[27] : HOME=/home/ap-73
Environ[28] : USERNAME=ap-73
Environ[29] : IM_CONFIG_PHASE=1
Environ[30] : LANG=en_US.UTF-8

1:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=
;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.
io=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31
:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.x
f=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;
5:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*
f=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35
yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00
36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.op
Environ[32] : XDG_CURRENT_DESKTOP=Unity
Environ[33] : WAYLAND_DISPLAY=wayland-0
Environ[34] : GIT_ASKPASS=/snap/code/126/usr/share/code/resources/app/extensions/git/dist/
Environ[35] : INVOCATION_ID=2b78ad4c25424ee3b523c43f600b7a5d
Environ[36] : MANAGERPID=1486
Environ[37] : CHROME_DESKTOP=code-url-handler.desktop
Environ[38] : GJS_DEBUG_OUTPUT=stderr
Environ[39] : VSCODE_GIT_ASKPASS_EXTRA_ARGS=--ms-enable-electron-run-as-node
Environ[40] : GNOME_SETUP_DISPLAY=:1
Environ[41] : LESSCLOSE=/usr/bin/lesspipe %s %s
Environ[42] : XDG_SESSION_CLASS=user
Environ[43] : TERM=xterm-256color
Environ[44] : GTK_PATH=/snap/code/126/usr/lib/x86_64-linux-gnu/gtk-3.0
Environ[45] : LESSOPEN=| /usr/bin/lesspipe %s
Environ[46] : USER=ap-73
Environ[47] : VSCODE_GIT_IPC_HANDLE=/run/user/1000/vscode-git-a43cdca4cd.sock
Environ[48] : DISPLAY=:0
Environ[49] : SHLVL=1
Environ[50] : LOCPATH=/snap/code/126/usr/lib/locale
Environ[51] : QT_IM_MODULE=ibus
Environ[52] : XDG_RUNTIME_DIR=/run/user/1000
Environ[53] : VSCODE_GIT_ASKPASS_MAIN=/snap/code/126/usr/share/code/resources/app/extensio
Environ[54] : JOURNAL_STREAM=8:33992
Environ[55] : XDG_DATA_DIRS=/home/ap-73/snap/code/126/.local/share:/home/ap-73/snap/code/1
```

Program – 5

Object -Write a program to print current (soft) limit and maximum (Hard) limits of all resources.

Code :

```
#include <stdio.h>      // Standard Input/Output library
#include <stdlib.h>     // Standard library
#include <errno.h>      // Error handling library
#include <sys/resource.h>    // System resource limits library

#define NUM_RESOURCES RLIM_NLIMITS   // Define a constant to represent the number
of system resources available

int main() {
   struct rlimit rl;   // Declare a structure to hold the resource limits
   int i;
   char* resource_names[NUM_RESOURCES] = {   // Declare an array of strings to hold
the names of the system resources
      "RLIMIT_CPU",
      "RLIMIT_FSIZE",
      "RLIMIT_DATA",
      "RLIMIT_STACK",
      "RLIMIT_CORE",
      "RLIMIT_RSS",
      "RLIMIT_NPROC",
      "RLIMIT_NOFILE",
      "RLIMIT_MEMLOCK",
      "RLIMIT_AS",
      "RLIMIT_LOCKS"
   };

   // Print the header row for the output table
   printf("\n%-15s\t%-15s\t%-15s\n", "Resource Name", "Soft Limit", "Hard Limit");

   // Loop through each system resource and retrieve its limits
   for (i = 0; i < NUM_RESOURCES; i++) {
      if (getrlimit(i, &rl) != 0) {   // Get the limits for the current system resource
         perror("getrlimit");    // Print an error message if the limits cannot be retrieved
         exit(EXIT_FAILURE);     // Exit the program with an error code
      }
```

```
        // Print the resource name and its corresponding soft and hard limits
        printf("%-15s\t%-15lld\t%-15lld\n", resource_names[i], (long long) rl.rlim_cur, (long long) rl.rlim_max);
    }

    return 0;   // Return 0 to indicate success
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_5$ ./prog5

Resource Name   Soft Limit      Hard Limit
RLIMIT_CPU      -1              -1
RLIMIT_FSIZE    -1              -1
RLIMIT_DATA     -1              -1
RLIMIT_STACK    8388608         -1
RLIMIT_CORE     0               -1
RLIMIT_RSS      -1              -1
RLIMIT_NPROC    46643           46643
RLIMIT_NOFILE   1048576         1048576
RLIMIT_MEMLOCK  1538584576      1538584576
RLIMIT_AS       -1              -1
RLIMIT_LOCKS    -1              -1
(null)          46643           46643
(null)          819200          819200
(null)          0               0
(null)          0               0
(null)          -1              -1
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_5$
```

Program – 6

Object -Write a program with an exit handler that outputs CPU usage.

Code :

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

int main(void) {
    time_t start, end; // declare two variables of type "time_t" to hold the start and end times
    start = time(NULL); // get the current time and store it in the "start" variable
    printf("Start time: %ld\n", start); // print the start time
    sleep(5); // sleep for 5 seconds
    end = time(NULL); // get the current time again and store it in the "end" variable
    printf("End time: %ld\n", end); // print the end time
    printf("Time slept: %ld seconds\n", end - start); // calculate and print the time slept
    return 0; // return success code
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_6$ ./prog6_gpt
Start time: 1682936731
End time: 1682936736
Time slept: 5 seconds
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_6$
```

Program – 7

Object -Write a program that prints it's & it's parent's process ID.

Code :

```
#include<stdio.h> // Including the standard input/output library header file
#include <unistd.h> // Including the POSIX library header file

int main (void) // Beginning of the main function
{
   // Printing the process ID of the current process
    printf("I am process %ld\n", (long)getpid());
  // Printing the parent process ID of the current process
    printf("My parent is %ld\n", (long)getppid());
    return 0; // Returning 0 to indicate successful termination of the program
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_7$ ./prog7
I am process 15046
My parent is 15014
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_7$
```

Program – 8

Object -Write a program that prints out various user & group ID's.

Code :

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

int main(void)
{
   // Print the real user ID of the current process
   printf("My real user ID is %5ld\n", (long)getuid());

   // Print the effective user ID of the current process
   printf("My effective user ID is %5ld\n", (long)geteuid());

   // Print the real group ID of the current process
   printf("My real group ID is %5ld\n", (long)getgid());

   // Print the effective group ID of the current process
   printf("My effective group ID is %5ld\n", (long)getegid());

   return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_8$ ./prog8
My real user ID is  1000
My effective user ID is  1000
My real group ID is  1000
My effective group ID is  1000
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_8$
```

Program – 9

Object --Write a program which uses fork to create a child process& then parent & child print their respective process ID's

Code :

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

int main() {
   pid_t pid; // declare a variable to store the process ID returned by fork()

   pid = fork(); // create a child process and store the process ID in the pid variable

   if (pid < 0) { // check if fork() failed
      fprintf(stderr, "Fork failed\n");
      exit(1);
   }
   if (pid == 0) { // code for child process
      printf("\nI am the child process, my pid is %d\n", getpid());
       exit(0); //terminate the child process
   }
   else { // code for parent process
      printf("I am the parent process, my pid is %d", getpid());
   }

   return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_9$ ./prog9_gpt
I am the parent process, my pid is 15542
I am the child process, my pid is 15543
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_9$
```

Program – 10

Object -Write a program that creates a chain of n processes, where n is a command line argument.

Code :

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

int main(int argc, char *argv[]) {
  int n, i;
  pid_t pid;
  // check if the number of command line arguments is correct
  if (argc != 2) {
    printf("Usage: %s n\n", argv[0]);
    exit(1);
  }
  // convert the command line argument to an integer
  n = atoi(argv[1]);
  // create the chain of n processes
  for (i = 0; i < n; i++) {
        pid = fork();
        if (pid < 0) {
        printf("Error: fork failed\n");
        exit(1);
        }
        if (pid == 0) {
        printf("Child process %d created, with parent process %d\n", getpid(), getppid());
        while (getppid() != 1);
        printf("Child process %d has completed, with parent process %d\n", getpid(),
getppid());
        exit(0);
        }
  }
//   // parent process
//   printf("Parent process exiting\n");
  exit(0);
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_10$ ./prog10_gpt 6
Child process 15919 created, with parent process 1486
Child process 15918 created, with parent process 1486
Child process 15915 created, with parent process 1486
Child process 15916 created, with parent process 1486
Child process 15914 created, with parent process 1486
Child process 15917 created, with parent process 1486
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_10$
```

Program – 11

Object -Write a program that creates a fan of n processes where n is passed as a command line argument.

Code :

```
#include <stdio.h>      //include standard input/output library
#include <stdlib.h>     //include standard library for functions such as atoi()
#include <unistd.h>     //include library for functions such as fork()

int main(int argc, char *argv[]) {  //start of main function, takes command line arguments
as inputs
   //declare and initialize process ID variable for child process
   pid_t childpid = 0;
   //declare variables for loop iteration and number of processes
   int i, n;
   //check if there is only one command line argument
   if (argc != 2) {
      fprintf(stderr, "Usage: %s processes\n", argv[0]);  //print error message to stderr
      return 1;          //exit program with error status
   }
    //convert command line argument to integer and assign to variable n
   n = atoi(argv[1]);
   for (i = 1; i <= n; i++) {   //iterate through loop until i is equal to n
   //if fork() returns a value less than or equal to 0, break out of loop
      if ((childpid = fork()) <= 0) {
         break;
      }
      //print process information to stderr
      fprintf(stderr, "i:%d process ID:%ld parent ID:%ld child ID:%ld\n",
         i, (long)getpid(), (long)getppid(), (long)childpid);
   }
   return 0;   //exit program with success status
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_11$ ./prog11 4
i:1 process ID:16178 parent ID:16136 child ID:16179
i:2 process ID:16178 parent ID:16136 child ID:16180
i:3 process ID:16178 parent ID:16136 child ID:16181
i:4 process ID:16178 parent ID:16136 child ID:16182
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_11$
```

Program – 12

Object -Write a program to show that same opened file can be shared by both parent and child processes.

Code :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main() {
    int fd;         // file descriptor for the opened file
    pid_t pid;      // process ID variable

    // open file in write mode, create it if it doesn't exist, with permissions 0644
    fd = open("file.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd < 0) {   // check if file was successfully opened
        perror("Error opening file");   // display error message if open() failed
        exit(1);    // exit the program with status code 1
    }

    // fork a child process
    pid = fork();   // create a copy of the current process
    if (pid < 0) {  // check if fork() failed
        perror("Error forking process");    // display error message if fork() failed
        exit(1);    // exit the program with status code 1
    }
    else if (pid == 0) {    // child process
        // write to the opened file using the file descriptor fd
        char *str = "This is the child process.\n";
        write(fd, str, strlen(str));   // write string to file
        printf("Child process wrote to file.\n");
    }
    else {  // parent process
        // write to the opened file using the file descriptor fd
        char *str = "This is the parent process.\n";
        write(fd, str, strlen(str));   // write string to file
        printf("Parent process wrote to file.\n");
    }
```

```
    // close file descriptor
    close(fd);  // close the file

    return 0;   // exit the program with status code 0 (success)
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_12$ ./prog12_g
Parent process wrote to file.
Child process wrote to file.
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_12$
```

OS_Programs > Practical_12 > file.txt

```
This is the parent process.
This is the child process.
```

Program – 13

Object -Write a program that creates a child process to run ls – l.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(void)
{
   pid_t childpid;
   // Call fork() to create a new child process
   childpid = fork();
   // Check if fork() failed and print an error message if it did
   if (childpid == -1)
   {
      perror("Failed to fork");
      return 1;
   }
   // If fork() returns 0, the current process is the child process
   if (childpid == 0)
   {
      // Execute the "ls -l" command in the child process using execl()
      execl("/bin/ls", "ls", "-l", NULL); // man 2 exec
      // If execl() returns, there was an error, so print an error message and return failure
      perror("Child failed to exec ls");
      return 1;
   }
   // If fork() returns a positive value, the current process is the parent process
   // Wait for the child process to complete using wait() and check for errors
   if (childpid != wait(NULL))
   {
      perror("Parent failed to wait due to signal or error");
      return 1;
   }
   return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_13$ ./prog_13
total 20
-rwxrwxrwx 1 root root 16088 May  1 16:36 prog_13
-rwxrwxrwx 1 root root  1177 May  1 16:36 prog_13.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_13$
```

Program – 14

Object -Write a program to create a zombie child and find its status using system (ps) command.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<signal.h>
int main(){
   int fd;
   // Creating a child process using fork()
   if((fd=fork())<0){
      printf("error in creating child");
      exit(1);
   }
   // Child process code: send a signal to itself to terminate
   if(fd==0)
      kill(getpid(),SIGKILL);
   // Parent process code: wait for 2 seconds
   else
      sleep(2);
   // Print process information using the "ps" command
   system("ps -f");
   return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_14$ ./prog_14
UID          PID    PPID  C STIME TTY          TIME CMD
ap-73      17092   11155  0 16:44 pts/36   00:00:00 /usr/bin/bash --init-file /snap/code/126/usr/
ap-73      17110   17092  0 16:44 pts/36   00:00:00 ./prog_14
ap-73      17111   17110  0 16:44 pts/36   00:00:00 [prog_14] <defunct>
ap-73      17123   17110  0 16:44 pts/36   00:00:00 sh -c ps -f
ap-73      17124   17123  0 16:44 pts/36   00:00:00 ps -f
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_14$
```

Program – 15

Object -Write a program to copy a file.

Code :

```
// Function to copy a file fromfd to tofd
#include <errno.h>
#include <unistd.h>
#define BLKSIZE 1024

int copyfile(int fromfd, int tofd) {
   char *bp;
   char buf[BLKSIZE];
   int bytesread;
   int byteswritten = 0;
   int totalbytes = 0;

   for (;;) {  // Infinite loop until end of file or error occurs
      // Read up to BLKSIZE bytes from fromfd
      while (((bytesread = read(fromfd, buf, BLKSIZE)) == -1) &&
            (errno == EINTR)) ;  // Handle interruption by signal

      if (bytesread <= 0)  // Real error or end-of-file on fromfd
         break;

      bp = buf;
      while (bytesread > 0) {
         // Write up to bytesread bytes to tofd
         while (((byteswritten = write(tofd, bp, bytesread)) == -1) &&
               (errno == EINTR)) ;  // Handle interruption by signal

         if (byteswritten < 0)  // Real error on tofd
            break;

         totalbytes += byteswritten;
         bytesread -= byteswritten;
         bp += byteswritten;
      }

      if (byteswritten == -1)  // Real error on tofd
         break;
```

```
    }

    return totalbytes;
}

// Main program to copy a file
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#define READ_FLAGS O_RDONLY
#define WRITE_FLAGS (O_WRONLY | O_CREAT | O_EXCL)
#define WRITE_PERMS (S_IRUSR | S_IWUSR)

int main(int argc, char *argv[]) {
    int bytes;
    int fromfd, tofd;

    // Check for correct number of arguments
    if (argc != 3) {
        fprintf(stderr, "Usage: %s from_file to_file\n", argv[0]);
        return 1;
    }

    // Open input file for reading
    if ((fromfd = open(argv[1], READ_FLAGS)) == -1) {
        perror("Failed to open input file");
        return 1;
    }

    // Create output file for writing
    if ((tofd = open(argv[2], WRITE_FLAGS, WRITE_PERMS)) == -1) {
        perror("Failed to create output file");
        return 1;
    }

    // Copy the contents of the input file to the output file
    bytes = copyfile(fromfd, tofd);

    // Print the number of bytes copied and the source and destination file names
    printf("%d bytes copied from %s to %s\n", bytes, argv[1], argv[2]);
```

```
    // Close the input and output files
    return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_15$ cat file1.txt
This is the text to be copied.
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_15$ ./prog15 file1.txt file2.tx
31 bytes copied from file1.txt to file2.txt
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_15$ cat file2.txt
This is the text to be copied.
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_15$
```

Program – 16

Object -Write a program for which output is automatically directed to a named file rather than on to the console.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
#include <unistd.h>
int main()
{
   int fd;
   // Open the file "test1" for writing or create it if it doesn't exist
   if((fd = open("test1", O_WRONLY|O_CREAT)) < 0){
      printf("Error in opening file..\n");
      exit(1);
   }
   // Close standard output (file descriptor 1) to redirect the output
   close(1);
   // Duplicate file descriptor fd to 1 (stdout)
   dup(fd);
   // The next line will be written to the file "test1" instead of the console
   printf("New Fun");
   // Close the file descriptor fd and standard output
   close(fd);
   return (0);
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_16$ g++ -o prog16 prog_16.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_16$ ./prog16
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_16$
```

```
OS_Programs > Practical_16 > test1
1    New Fun
```

Program – 17

Object -Write a program that redirects standards output to the file my.file (or Write a program that do the following operation cat XYZ > myfile). using dup2 rather than dup.

Code :

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>

#define CREATE_FLAGS (O_WRONLY | O_CREAT | O_APPEND) // Define flags for file creation
#define CREATE_MODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH) // Define access permissions for the file

int main(void){
    int fd; // file descriptor

    // Open or create file "my.file" with the flags and mode specified
    fd = open("my.file", CREATE_FLAGS, CREATE_MODE);
    if (fd == -1){
        perror("Failed to open my.file"); // Print error message if failed to open the file
        return 1; // Exit program with error code
    }
    // Duplicate the file descriptor to the standard output file descriptor
    if (dup2(fd, STDOUT_FILENO) == -1){  //dup2
        perror("Failed to redirect standard output"); // Print error message if failed to redirect standard output
        return 1; // Exit program with error code
    }
    // Close the file descriptor that was duplicated
    if (close(fd) == -1){
        perror("Failed to close the file"); // Print error message if failed to close the file
        return 1; // Exit program with error code
    }
    // Write "OK" to the standard output file descriptor
    if (write(STDOUT_FILENO, "OK", 2) == -1){
        perror("Failed in writing to file"); // Print error message if failed to write to file
        return 1; // Exit program with error code
    }
```

```
    return 0; // Exit program with success code
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_17$ g++ -o prog_17 prog_17.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_17$ ./prog_17
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_17$
```

OS_Programs > Practical_17 > my.file

```
1    OK
```

Program – 18

Object -Write a program to create an empty directory using system calls

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/stat.h>

// Main function takes two arguments, argc (argument count) and argv (argument vector)
main(int argc, char *argv[])
{
   // Check if the argument count is not equal to 2
   if(argc!=2)
   {
      printf("Usages: ./a.out directory"); // Print error message for incorrect usage
      exit(1); // Exit the program with error code
   }

   // Create a directory with the specified name and access permissions
   if(mkdir(argv[1],744)!=0)
      printf("Error in Making Directory"); // Print error message if the directory creation
fails
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_18$ g++ -o prog_18 prog_18.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_18$ ./prog_18 emptyDir
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_18$ sudo ls -ld emptyDir/
[sudo] password for ap-73:
drwxrwxrwx 1 root root 0 May  1 17:31 emptyDir/
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_18$
```

Program – 19

Object - Write a program to remove a directory using system call.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>

// Main function takes two arguments, argc (argument count) and argv (argument vector)
int main(int argc, char *argv[])
{
   // Check if the argument count is not equal to 2
   if(argc!=2)
   {
      fprintf(stderr,"Too Less Arguments"); // Print error message for incorrect usage
      exit(1); // Exit the program with error code
   }

   // Remove the directory with the specified name
   if(remove(argv[1])!=0)
      fprintf(stderr,"Error in Removing Directory"); // Print error message if the directory
removal fails
   exit(1); // Exit the program with error code
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_19$ ./prog_18 dir1
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_19$ g++ -o ./prog_19 prog_19.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_19$ ./prog_19 dir1/
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_19$ sudo ls -ld dir1
[sudo] password for ap-73:
ls: cannot access 'dir1': No such file or directory
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_19$
```

Program – 20

Object - Write a program to output current working directory.

Code :

```
#include <limits.h>
#include <stdio.h>
#include <unistd.h>

// Define PATH_MAX to be 255 if it is not already defined
#ifndef PATH_MAX
#define PATH_MAX 255
#endif

int main(void)
{
   char mycwd[PATH_MAX];
   // Get the current working directory and store it in mycwd array
   if (getcwd(mycwd, PATH_MAX) == NULL)
   {
      perror("Failed to get current working directory"); // Print error message if getcwd
fails
      return 1; // Exit the program with error code
   }
   // Print the current working directory
   printf("Current working directory: %s\n", mycwd);
   return 0; // Exit the program
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_20$ g++ -o prog_20 prog_20.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_20$ ./prog_20
Current working directory: /mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_20
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_20$
```

Program – 21

Object - Write a program to list files in a directory.

Code :

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    struct dirent *direntp; // Pointer to directory entry
    DIR *dirp; // Pointer to directory stream
    if (argc != 2) {
        fprintf(stderr, "Usage: %s directory_name\n", argv[0]);
        return 1;
    }
    // Attempt to open the specified directory
    if ((dirp = opendir(argv[1])) == NULL) {
        perror ("Failed to open directory"); // Print error message if failed
        return 1;
    }
    // Read and print each directory entry
    while ((direntp = readdir(dirp)) != NULL) {
        printf("%s\n", direntp->d_name);
    }
    // Close the directory stream and check for errors
    while ((closedir(dirp) == -1) && (errno == EINTR)) ;
    return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_21$ g++ -o prog_21 prog_21.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_21$ ./prog_21 dir
.
..
test1
test2
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_21$
```

Program – 22

Object - Write a program that returns true if a given file is a directory & false otherwise.

Code :

```
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>

int main(int argc, char *argv[]) {
   struct stat statbuf; // Struct to hold file/directory information
   // Attempt to get status information for the specified file/directory
   if (stat(argv[1], &statbuf) == -1) {
      perror ("Failed to get status of file/directory"); // Print error message if failed
      return 1;
   }
   else {
      // If successful, determine if the specified path is a directory or a file using S_ISDIR
macro
      if (S_ISDIR(statbuf.st_mode)) {
         printf("%s : is a directory\n",argv[1]);
      }
      else {
         printf("%s : is a file\n",argv[1]);
      }
   }
   return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_22$ g++ -o prog_22 prog_22.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_22$ ./prog_22 dir
dir : is a directory
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_22$ ./prog_22 test
test : is a file
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_22$
```

Program – 23

Object - Write a program that can display the type of a given file like regular, directory etc.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>

main(int argc, char *argv[]) {
   struct stat statbuff; // Struct to hold file/directory information
   int check; // Return value of stat() function

   if(argc != 2) {
      printf("Can accept only two arguments"); // Print error message if incorrect number of
arguments is passed
      exit(1);
   }
   // Get status information for the specified file/directory using stat() function
   check = stat(argv[1], &statbuff);

   // Check if stat() was successful
   if(check == 0) {
      // Determine the type of the specified file/directory using S_IS* macros
      if(S_ISREG(statbuff.st_mode)) {
         printf("Regular File");
      }
      else if(S_ISDIR(statbuff.st_mode)) {
         printf("Directory");
      }
      else if(S_ISCHR(statbuff.st_mode)) {
         printf("Character Device");
      }
      else {
         printf("Other File");
      }
   }
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_23$ g++ -o prog_23 prog_23.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_23$ ./prog_23 dir
Directoryap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_23$ ./prog_23 test
Regular Fileap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_23$
```

Program – 24

Object - Write a program to display the permission of a given file.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>

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

    struct stat statbuff; // Struct to hold file/directory information
    int check; // Return value of stat() function

    if(argc != 2) {
        printf("Can Accept only two arguments"); // Print error message if incorrect number of
arguments is passed
        exit(1);
    }

    // Get status information for the specified file/directory using stat() function
    check = stat(argv[1], &statbuff);

    // Check if stat() was successful
    if(check == 0) {
        // Check permission for Owner
        if((statbuff.st_mode & S_IRUSR) == S_IRUSR) {
            printf("Owner has Read Permission\n");
        }
        if((statbuff.st_mode & S_IWUSR) == S_IWUSR) {
            printf("Owner has Write Permission\n");
        }
        if((statbuff.st_mode & S_IXUSR) == S_IXUSR) {
            printf("Owner has Execute Permission\n");
        }

        // Check permission for Group
        if((statbuff.st_mode & S_IRGRP) == S_IRGRP) {
            printf("Group has Read Permission\n");
```

```
        }
        if((statbuff.st_mode & S_IWGRP) == S_IWGRP) {
            printf("Group has Write Permission\n");
        }
        if((statbuff.st_mode & S_IXGRP) == S_IXGRP) {
            printf("Group has Execute Permission\n");
        }

        // Check permission for Others
        if((statbuff.st_mode & S_IROTH) == S_IROTH) {
            printf("Others has Read Permission\n");
        }
        if((statbuff.st_mode & S_IWOTH) == S_IWOTH) {
            printf("Others has Write Permission\n");
        }
        if((statbuff.st_mode & S_IXOTH) == S_IXOTH) {
            printf("Others has Executed Permission\n");
        }
    }
}
```

Output :

```
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_24$ g++ -o prog_24 prog_24.c
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_24$ ./prog_24 test
  Owner has Read Permission
  Owner has Write Permission
  Owner has Execute Permission
  Group has Read Permission
  Group has Write Permission
  Group has Execute Permission
  Others has Read Permission
  Others has Write Permission
  Others has Executed Permission
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_24$ ./prog_24 dir
  Owner has Read Permission
  Owner has Write Permission
  Owner has Execute Permission
  Group has Read Permission
  Group has Write Permission
  Group has Execute Permission
  Others has Read Permission
  Others has Write Permission
  Others has Executed Permission
○ ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_24$
```

Program – 25

Object - Write a program to execute the equivalent of ls –l | sort –n +4.

Code :

```
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void){
   pid_t childpid; //Variable to hold child process ID
   int fd[2]; //Array to hold read and write file descriptors

   //Create pipe and fork child process
   if ((pipe(fd) == -1) || ((childpid = fork()) == -1))
   {
      perror("Failed to setup pipeline"); //Print error message if pipe or fork fails
      return 1;
   }

   //Child process for ls command
   if (childpid == 0)
   {
      //Redirect the standard output of child to the write end of the pipe
      if (dup2(fd[1], STDOUT_FILENO) == -1)
         perror("Failed to redirect stdout of ls");
      //Close read and write file descriptors not required by child
      else if ((close(fd[0]) == -1) || (close(fd[1]) == -1))
         perror("Failed to close extra pipe descriptors on ls");
      else
      {
         //Execute ls command with options "-l" to list files in long format
         execl("/bin/ls", "ls", "-l", NULL);
         perror("Failed to exec ls"); //Print error message if execution of ls fails
      }
      return 1;
   }

   //Parent process for sort command
   if (dup2(fd[0], STDIN_FILENO) == -1)
```

```
        perror("Failed to redirect stdin of sort"); //Redirect the standard input of parent to the
read end of the pipe
    //Close read and write file descriptors not required by parent
    else if ((close(fd[0]) == -1) || (close(fd[1]) == -1))
        perror("Failed to close extra pipe file descriptors on sort");
    else
    {
        //Execute sort command with options "-n" to sort numerically and "+4" to skip first
four fields of input
        execl("/bin/sort", "sort", "-n", "+4", NULL);
        perror("Failed to exec sort"); //Print error message if execution of sort fails
    }
    return 1;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_25$ g++ -o prog_25 prog_25.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_25$ ./prog_25
total 20
-rwxrwxrwx 1 root root  1873 May  1 18:50 prog_25.c
-rwxrwxrwx 1 root root 16224 May  1 18:53 prog_25
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_25$
```

Program – 26

Object - Write a program to handle SIGUSR1 and SIGUSR2 signal.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<signal.h>
#include<unistd.h>

void fun(int); // declaration of signal handler function

int main()
{
   char a[200];

   // register signal handler for SIGUSR1
   if((signal(SIGUSR1,fun))==SIG_ERR)
   {
      printf("Handler not registered\n");
      exit(1);
   }

   // register signal handler for SIGUSR2
   if((signal(SIGUSR2,fun))==SIG_ERR)
   {
      printf("Handler not registered\n");
      exit(1);
   }

   // enter an infinite loop and wait for signals
   while(1)
      pause(); // pause the program until a signal is received
}

// implementation of signal handler function
void fun(int i)
{
```

```
    if(i==SIGUSR1)
    {
        printf("SIGUSR1 INTRRUPT");
        fflush(NULL); // flush output buffer to print message immediately
    }
    else if(i==SIGUSR2)
    {
        printf("SIGUSR2 INTRRUPT");
        fflush(NULL);
    }
    //raise(SIGKILL); // send SIGKILL signal to terminate program
}
```

Output :

```
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$ ./prog_26 &
  [1] 24164
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$ ps
      PID TTY          TIME CMD
    23804 pts/50   00:00:00 bash
    24164 pts/50   00:00:00 prog_26
    24184 pts/50   00:00:00 ps
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$ kill -s SIGUSR1 24164
○ ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$ SIGUSR1 INTRRUPT
● ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$ kill -s SIGUSR2 24164
○ SIGUSR2 INTRRUPTap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_26$
```

Program – 27

Object - Write a program which suspends itself till it receives a SIGALARM signal.

Code :

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

void sig_alrm(int); // declaration of signal handler function

int main(int argc, char *argv[])
{
   // register signal handler for SIGALRM
   if((signal(SIGALRM, sig_alrm))==SIG_ERR)
      printf("Not Registered");
   // set a timer for 5 seconds using alarm()
   alarm(5);
   // pause the program until a signal is received
   pause();
   return 0;
}
// implementation of signal handler function
void sig_alrm(int sig)
{
   // check if the signal received is SIGALRM
   if(sig==SIGALRM)
      printf("Wake Up\n");
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_27$ g++ -o prog_27 prog_27.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_27$ ./prog_27
Wake Up
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_27$
```

Program – 28

Object - Write a program which prints the second's part of current time whenever the SIGALRM signal is received by the program.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<signal.h>
#include<time.h>
#include<unistd.h>

void sig_hand(int); // declaration of signal handler function
int main(){
   int i=1;
   pid_t pid;
   // register the signal handler
   if(signal(SIGALRM,sig_hand)==SIG_ERR)
      printf("Not Registered");
   while(i<=5){ // loop to send signals after 2 seconds delay
      i++;
      pid=getpid();
      sleep(2); // wait for 2 seconds
      kill(pid,SIGALRM); // send the signal
   }
   return 0;
}
void sig_hand(int sig)
{
   struct tm *t; // structure to hold the broken-down time
   time_t tt; // variable to hold the time in seconds
   if(sig==SIGALRM) // check if the received signal is SIGALRM{
      tt=time(NULL); // get the current time in seconds since epoch
      t=localtime(&tt); // convert the time to broken-down time
      printf("%d\n",t->tm_sec); // print the seconds field of the time
   }
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_28$ g++ -o prog_28 prog_28.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_28$ ./prog_28
31
33
35
37
39
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_28$
```

Program – 29

Object - Write a Program to print the entries of passwd file for a given user name or user ID.

Code :

```
#include<pwd.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<signal.h>
#include<time.h>
#include<error.h>
#include<ctype.h>

int main(){
   char u_name[10]; // variable to hold the username entered by user
   char ch; // variable to hold the user's choice of input
   uid_t u_id; // variable to hold the user ID entered by user
   struct passwd *p; // pointer to the passwd structure

   printf("Enter Your Choice\n");
   printf("Whether you want to enter UNAME or UID?(N or I)"); // prompt user for choice
of input
   scanf("%c",&ch); // read the user's choice

   if((ch == 'N')|| (ch == 'n')) // if user chooses to enter username
   {
      printf("Enter UNAME");
      scanf("%s",u_name); // read the username entered by user
      p=getpwnam(u_name); // get the passwd structure for the given username
      printf("\n%s\n %s\n %d\n %d\n %s\n %s\n %s\n", p->pw_name, p->pw_passwd, p-
>pw_uid,p->pw_gid,p->pw_gecos, p->pw_dir, p->pw_shell); // print the information
retrieved from passwd structure
   }
   else if((ch == 'I' || 'i')) // if user chooses to enter user ID
   {
      printf("Enter UID");
      scanf("%d",&u_id); // read the user ID entered by user
```

```
    p= getpwuid (u_id); // get the passwd structure for the given user ID
    printf("\n%s\n %s\n %d\n %d\n %s\n %s\n %s\n", p->pw_name, p->pw_passwd, p->pw_uid,p->pw_gid,p->pw_gecos, p->pw_dir, p->pw_shell); // print the information retrieved from passwd structure
  }
  else
    printf("Wrong Choice"); // if user enters a wrong choice
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_29$ g++ -o prog_29 prog_29.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_29$ ./prog_29
Enter Your Choice
Whether you want to enter UNAME or UID?(N or I)N
Enter UNAMEap-73

ap-73
 x
 1000
 1000
 Abhishek,,,
 /home/ap-73
 /bin/bash
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_29$ ./prog_29
Enter Your Choice
Whether you want to enter UNAME or UID?(N or I)I
Enter UID1000

ap-73
 x
 1000
 1000
 Abhishek,,,
 /home/ap-73
 /bin/bash
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_29$
```

Program – 30

Object - Write a program to print the details of the system.

Code :

```
#include<grp.h>    // header file that provides access to group-related functions
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>

int main()    // the main function
{
   char g_name[10];
   gid_t gid;
   char ch;
   struct group *g;    // a pointer to a group structure

   printf("Enter Your Choice: \nEnter Group Name(N) \nEnter Group ID (I)\n");
   printf("Enter Choice");
   scanf("%c",&ch);    // prompts the user to enter their choice

   switch(ch)    // evaluates the user's choice
   {
      case 'N':
      case 'n':
         printf("Enter GNAME:");
         scanf("%s",g_name);    // reads the group name entered by the user
         g=getgrnam(g_name);    // retrieves the group information using the group name
         printf("\n %s %s %d\n", g->gr_name, g->gr_passwd, g->gr_gid);    // prints the
group information
         break;

      case 'I':
      case 'i':
         printf("Enter GID:");
         scanf("%d",&gid);    // reads the group ID entered by the user
         g=getgrgid(gid);    // retrieves the group information using the group ID
         printf("\n %s %s %d\n", g->gr_name, g->gr_passwd, g->gr_gid);    // prints the
group information
         break;
```

```
        default:
            printf("Wrong Choice");
    }
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_30$ g++ -o prog_30 prog_30.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_30$ ./prog_30
Enter Your Choice:
Enter Group Name(N)
Enter Group ID (I)
Enter ChoiceN
Enter GNAME:ap-73

 ap-73 x 1000
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_30$ ./prog_30
Enter Your Choice:
Enter Group Name(N)
Enter Group ID (I)
Enter ChoiceI
Enter GID:1000

 ap-73 x 1000
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_30$
```

Program – 31, 32

Object – Write two programs :

1. Reads what is written to a named pipe & writes it to standard output.
2. Write an informative message to a named pipe.

Code 1 :

```
// server
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>

#define BLKSIZE 1024
#define FIFOARG 1
#define FIFO_PERMS (S_IRWXU | S_IWGRP| S_IWOTH)

// function prototype
int copyfile(int fromfd, int tofd);

int main(int argc, char *argv[]) {
    int requestfd;
    if (argc != 2) { /* name of server fifo is passed on the command line */
        fprintf(stderr, "Usage: %s fifoname > logfile\n", argv[0]);
        return 1;
    }

    /* create a named pipe to handle incoming requests */
    if ((mkfifo(argv[FIFOARG], FIFO_PERMS) == -1) && (errno != EEXIST)) {
        perror("Server failed to create a FIFO");
        return 1;
    }

    /* open a read/write communication endpoint to the pipe */
    if ((requestfd = open(argv[FIFOARG], O_RDWR)) == -1) {
        perror("Server failed to open its FIFO");
        return 1;
    }
```

```
    // copy the data received from the named pipe to STDOUT
    copyfile(requestfd, STDOUT_FILENO);

    return 1;
}

// function to copy data from one file descriptor to another
int copyfile(int fromfd, int tofd) {
    char *bp;
    char buf[BLKSIZE];
    int bytesread;
    int byteswritten = 0;
    int totalbytes = 0;
    for (;;) { // infinite loop
        while (((bytesread = read(fromfd, buf, BLKSIZE)) == -1) && (errno == EINTR))
            ; // handle interruption by signal

        if (bytesread <= 0) // real error or end-of-file on fromfd
            break;

        bp = buf;
        while (bytesread > 0) {
            while (((byteswritten = write(tofd, bp, bytesread)) == -1) && (errno == EINTR))
                ; // handle interruption by signal
            if (byteswritten < 0) // real error on tofd
                break;

            totalbytes += byteswritten;
            bytesread -= byteswritten;
            bp += byteswritten;
        }

        if (byteswritten == -1) // real error on tofd
            break;
    }
    return totalbytes;
}
```

Code 2 :

```
// Client
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/stat.h>
#include <limits.h>

#define FIFOARG 1 // Argument index for the name of the server FIFO.

int main(int argc, char *argv[])
{
    time_t curtime;
    int len;
    char requestbuf[_PC_PIPE_BUF];
    int requestfd;
    if (argc != 2)
    { // Check if the argument count is correct.
        fprintf(stderr, "Usage: %s fifoname\n", argv[0]);
        return 1;
    }
    // Open the FIFO for writing.
    if ((requestfd = open(argv[FIFOARG], O_WRONLY)) == -1)
    {
        perror("Client failed to open log fifo for writing");
        return 1;
    }

    // Get the current time and format the message to be sent to the server.
    curtime = time(NULL);
    snprintf(requestbuf, _PC_PIPE_BUF, "%d: %s", (int)getpid(), ctime(&curtime));
    len = strlen(requestbuf);

    // Write the message to the server and check for errors.
    if (write(requestfd, requestbuf, len) != len)
    {
```

```
        perror("Client failed to write");
        return 1;
    }
    // Close the FIFO.
    close(requestfd);
    return 0;
}
```

Output :

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ g++ -o server prog_31.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ g++ -o client prog_32.c
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ ./server mypipe
28112817
```

```
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ ./client
Usage: ./client fifoname
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ ./client mypipe
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$ ./client mypipe
ap-73@AP:/mnt/A2A25781A257593D/Practical6th/OS_Programs/Practical_31&32$
```

Index

Sr. no.	Program	Date	Remarks
1.	Write a program to display a file page wise assuming a page has 10 lines and each line has 80 characters		
2.	Write a Program which converts all the small case letters in a file into appropriate capital letters.		
3.	Write a program to print the details of the system (use uname sys call)		
4.	Write a program which will print the list of environment variable and also print the value of the PATH system variable		
5.	Write a program to print current (soft) limit and maximum (Hard) limits of all resources		
6.	Write a program with an exit handler that outputs CPU usage.		
7.	Write a program that prints it's & it's parent's process ID.		
8.	Write a program that prints out various user & group ID's.		
9	Write a program which uses fork to create a child process& then parent & child print their respective process ID's.		
10.	Write a program that creates a chain of n processes, where n is a command line argument.		
11.	Write a program that creates a fan of n processes where n is passed as a command line argument.		
12.	Write a program to show that same opened file can be shared by both parent and child processes		
13.	Write a program that creates a child process to run ls – l		

14.	Write a program to create a zombie child and find its status using system (ps) command		
15.	Write a program to copy a file.		
16.	Write a program for which output is automatically directed to a named file rather than on to the console.		
17.	Write a program that redirects standards output to the file my.file (or Write a program that do the following operation cat XYZ > myfile). using dup2 rather than dup.		
18.	Write a program to create an empty directory using system calls.		
19.	Write a program to remove a directory using system call.		
20.	Write a program to output current working directory.		
21.	Write a program to list files in a directory.		
22.	Write a program that returns true if a given file is a directory & false otherwise.		
23.	Write a program that can display the type of a given file like regular, directory etc.		
24.	Write a program to display the permission of a given file.		
25.	Write a program to execute the equivalent of ls –l | sort –n +4.		
26.	Write a program to handle SIGUSR1 and SIGUSR2 signal.		
27.	Write a program which suspends itself till it receives a SIGALARM signal.		
28.	Write a program which prints the second's part of current time whenever the SIGALRM signal is received by the program.		

29.	Write a Program to print the entries of passwd file for a given user name or user ID.		
30.	Write a program to print the details of the system.		
31.	Write a program which Reads what is written to a named pipe & writes it to standard output		
32.	Write an informative message to a named pipe.		