

Nama : Hudi Pradjanu
NIM : L200180128

Modul 8

Exec.c

The image shows a Linux desktop environment with a dark theme. In the top window, a terminal session is running under the user 'draph@hudt'. The user has run the command 'gcc exec.c' and then './a.out /bin/ls'. The terminal output shows:

```
draph@hudt:~$ gcc exec.c
draph@hudt:~$ ./a.out /bin/ls
Child Process
a.out      examples.desktop  glassfish-4.1.1  Pictures  Videos
Desktop   exec.c            Music        Public    walt
Documents fork              netbeans-8.2   snap      wait.c
Downloads fork.c           NewFolder     Templates
```

In the bottom window, a text editor titled 'exec.c' is open, displaying the C source code for a program that demonstrates process creation and execution. The code includes comments explaining the usage and the logic for spawning a child process and executing a specified command.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <walt.h>
int main(int argc, char*argv[]){
    pid_t pid;
    int i;
    if (argc!=3){
        printf("\nInsufficient arguments to load program");
        printf("\nUsage : ./a.out <path> <cmd>\n");
    }
    switch(pid = fork()){
    case -1:
        printf("Fork Failed");
        exit(-1);
    case 0:
        printf("Child Process\n");
        i = exec(argv[1], argv[2], NULL);
        if (i < 0){
            printf("program not loaded using exec system call\n", argv[2]);
            exit(-1);
        }
    default:
        wait(NULL);
        printf("Child Terminated\n");
        exit(0);
    }
}
```

Stat.c



The screenshot shows a terminal window with the following content:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/stat.h>
int main(int argc, char*argv[]){
    struct stat
    file; int n;
    if (argc != 2)
    {
        printf("Usage: ./a.out <filename>\n"); exit(-1);
    }
    if ((n = stat(argv[1], &file)) == -1)
    {
        perror(argv[1]);
        exit(-1);
    }
    printf("User id : %d\n", file.st_uid);
    printf("Group Id : %d\n", file.st_gid);
    printf("Block size: %d\n", file.st_blksize);
    printf("Block allocated: %d\n", file.st_blocks);
    printf("Inode no. : %d\n", file.st_ino);
    printf("Last accessed : %s", ctime(&(file.st_atime)));
    printf("Last modified : %s", ctime(&(file.st_mtime)));
    printf("File size : %bytes\n", file.st_size);
    printf("No. of links : %d\n", file.st_nlink);
    printf("Permission : ");
    printf( (S_ISDIR(file.st_mode)) ? "d" : "-");
    printf( (file.st_mode & S_IRUSR) ? "r" : "-");
    printf( (file.st_mode & S_IWUSR) ? "w" : "-");
    printf( (file.st_mode & S_IXUSR) ? "x" : "-");
    printf( (file.st_mode & S_IGRP) ? "r" : "-");
    printf( (file.st_mode & S_IWGRP) ? "w" : "-");
    printf( (file.st_mode & S_IXGRP) ? "x" : "-");
    printf( (file.st_mode & S_IROTH) ? "r" : "-");
    printf( (file.st_mode & S_IWOTH) ? "w" : "-");
    printf( (file.st_mode & S_IXOTH) ? "x" : "-");
    printf("\n");
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/stat.h>
int main(int argc, char* argv[])
{
    struct stat
    file; int n;
    if (argc != 2)
    {
        printf("Usage: ./a.out stat.c");
        exit(-1);
    }
    if ((n = stat(argv[1], &file)) == -1)
    {
        perror(argv[1]);
        exit(-1);
    }
    printf("User id : %d\n", file.st_uid);
    printf("Group id : %d\n", file.st_gid);
    printf("File size : %ld\n", file.st_size);
    printf("No. of links : %d\n", file.st_nlink);
    printf("Permission : %s\n", file.st_mode & S_ISDIR ? "d" : "-");
    printf("Block size: %ld\n", file.st_blksize);
    printf("Block allocated: %ld\n", file.st_blocks);
    printf("Inode no. : %ld\n", file.st_ino);
    printf("Last accessed: %s\n", ctime(&file.st_atime));
    printf("Last modified : %s\n", ctime(&file.st_mtime));
    printf("File type : %s\n", file.st_filetype == S_IFREG ? "Regular" :
    file.st_filetype == S_IFDIR ? "Directory" :
    file.st_filetype == S_IFLNK ? "Symbolic Link" :
    file.st_filetype == S_IFBLK ? "Character Device" :
    file.st_filetype == S_IFCHR ? "Block Device" :
    file.st_filetype == S_IFSOCK ? "Socket" :
    "Unknown");
}
```

Dirlist.c

A screenshot of a Linux desktop environment. At the top, there is a dock with icons for various applications like a browser, file manager, and terminal. Below the dock, there is a terminal window titled "Activities Terminal". The terminal shows the command "draph@hudl:~\$ gcc dirlist.c" followed by the output of the program. The program lists all files and directories in the current directory. To the right of the terminal, there is a file manager window titled "Activities Text Editor". It has two tabs: "stat.c" and "dirlist.c". The "dirlist.c" tab contains the C code for the program.

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
int main(int argc, char*argv[])
{
    struct dirent *dptr;
    DIR *dname;
    if (argc != 2)
    {
        printf("Usage: ./a.out <dirname>\n");
        exit(-1);
    }
    if ((dname = opendir(argv[1])) == NULL)
    {
        perror(argv[1]);
        exit(-1);
    }
    while (dptr=readdir(dname))
    {
        printf("%s\n", dptr->d_name);
    }
    closedir(dname);
}
```

A screenshot of a Linux desktop environment. At the top, there is a dock with icons for various applications like a browser, file manager, and terminal. Below the dock, there is a terminal window titled "Activities Terminal". The terminal shows the command "draph@hudl:~\$ gcc dirlist.c" followed by the output of the program. The program lists all files and directories in the current directory. To the right of the terminal, there is a file manager window titled "Activities Text Editor". It has two tabs: "stat.c" and "dirlist.c". The "dirlist.c" tab contains the C code for the program.

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
int main(int argc, char*argv[])
{
    struct dirent *dptr;
    DIR *dname;
    if (argc != 2)
    {
        printf("Usage: ./a.out <dirname>\n");
        exit(-1);
    }
    if ((dname = opendir(argv[1])) == NULL)
    {
        perror(argv[1]);
        exit(-1);
    }
    while (dptr=readdir(dname))
    {
        printf("%s\n", dptr->d_name);
    }
    closedir(dname);
}
```