PRAKTIKUM SISTEM OPERASI

MODUL 8

Disusun Oleh :

MUHAMMAD MIFTAHUL HUDA

L200210230

E

TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2022/2023

KEGIATAN PRAKTIKUM:

1. Membuat sebuah ‘child process’ (proses baru) dengan menggunakan system call ‘fork’. Mambuat program dengan algoritma sebagai berikut :
   (contoh program diberikan pada bagian berikutnya).
   a. Deklarasi sebuah variabel x yang akan diakses bersama antara child proses dan parent proses.
   b. Membuat sebuah child proses menggunakan system call fork.
   c. Jika return value bernilai -1, tampilkan teks ‘Pembuatan proses GAGAL’, dilanjutkan dengan keluar program dengan perintah system call ‘exit’.
   d. Jika return value sama dengan 0 (NOL), Tampilkan teks ‘Child Process’, tampilkanID proses dari child proses menggunakan perintah system call ‘getpid’, tampilkan nilai x, dan tampilkan ID proses parent dengan perintah system call ‘getppid’.
   e. Untuk nilai return value yang lainnya, tampilkan teks ‘Parent process’, tampilkan ID dari parent proses menggunakan perintah system call getpid, tampilkan nilai x, dan tampilkan ID dari proses shell menggunakan perintah system call getppid.
   f. Stop
   g. Kode Pogram & Output:

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

main() {
    pid_t pid;
    int x = 5;
    pid = fork();
    x++;

    if (pid < 0) {
        printf("Process creation error");
        exit(-1);
    } else if (pid == 0) {
        printf("\nChild process:");
        printf("\nProcess id is %d", getpid());
        printf("\nValue of x is %d", x);
        printf("\nProcess id of parent is %d\n\n", getppid());
    } else {
        printf("\nParent process:");
        printf("\nProcess id is %d", getpid());
        printf("\nValue of x is %d", x);
        printf("\nProcess id of shell is %d\n", getppid());
    }
}
```

input

```
    6 | main() {
      | ^~~~

Parent process:
Process id is 628
Value of x is 6
Process id of shell is 627


...Program finished with exit code 0
Press ENTER to exit console.
```

2. Menghentikan sementara (block) proses parent sampai dengan proses child selesai,menggunkaan perintah system call ‘wait’.
   Membuat program dengan algoritma sebagai berikut,contoh program diberikan pada bagian berikutnya.
   a. Membuat sebuah child proses menggunakan sytem call ‘fork’.
   b. Jika return value bernilai -1, selanjutnya tampilkan teks ‘pembuatan proses gagal’, dan kelaur program dengan menggunakan perintah system call ‘exit’.
   c. Jika return value berupa angka positif (> 0), ‘pause’ hentikan sementara ‘parent’ proses tunggu sampai child proses berakhir dengan menggunakan perintah system call ‘wait’. Tampilkan teks ‘Parent starts’, selanjutnya tampilkan nomor genap mulai dari 0 s/d 10, terakhir tampilkan teks ‘Parent end’.
   d. Jika return value bernilai 0 (NOL), tampilkan teks ‘Child start’, tampilkan nomor ganjil mulai dari 0 s/d 10, selanjutnya tampilkan teks ‘child ends’
   e. Stop
   f. Kode Program& Output:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
main() {
    int i, status;
    pid_t pid;
    pid = fork();

    if (pid < 0)
    {
        printf("\nPembuatan proses gagal\\n");
        exit(-1);
    }
    else if(pid > 0)
    {
        wait(NULL);
        printf ("\nParent starts\nNomor Genap:");
        for (i=2;i<=10;i+=2)
            printf ("%3d",i);
        printf("\nParent ends\n");
    }
    else if (pid == 6)
    {
        printf ("Child starts\nNomor Ganjil:");
        for (i=1;1<10;i+=2)
            printf ("%3d",i);
        printf ("\nChild ends\n");
    }
}
```

```
    6 | main() {
      | ^~~~

Parent starts
Nomor Genap:  2  4  6  8 10
Parent ends


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Loading program yang dapat dieksekusi dalam sebuah ‘child’ proses menggunakan perintah system call ‘exec’. Membuat program dengan algoritma sebagai berikut:
   (contoh program diberikan pada bagian berikutnya).
   a. Jika terdapat 3 argumen dalam command-line berhenti (stop).
   b. Membuat child proses dengan perintah system call ‘fork’
   c. Jika return value adalah -1, selanjuntya tampilkan teks ‘Pembuatan proses Gagal’, dan keluar program dengan perintah system call exit.
   d. Jika return value >0 (positif), selanjutnya hentikan parent-proses sementara hingga child-proses berakhir dengan menggunakan perintah system call wait. Tampilkan teks ‘Child berakhir’, dan hentikan parent-proses.
   e. Jika return value sama dengan 0 (NOL), selanjutnya tampilkan teks ‘Child starts’, load program dari lokasi yang diberikan dalam ‘path’ ke dalam child-proses, menggunakan perintah system call ‘exec’. Jika return value dari perintah ‘exec’ adalah bilangan negatif, tampilkan error yang terjadi dan stop. Hentikan child- proses.
   f. Stop
   g. Kode Program & Output:

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

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"); exit(-1);
    }

    switch(pid = fork()){
        case -1:
            printf("Fork failed");
            exit(-1);
        case 0:
            printf("Child process\n");
            i = execl(argv[1], argv[2], 0);
            if (i < 0){
                printf("%s program not loaded using exec system call\n", argv[2]);
                exit(-1);
            }
        default:
            wait(NULL);
            printf("Child Terminated\n");
            exit(0);
    }
}
```

input

```
    6 | main(int argc, char*argv[]) {
      | ^~~~
main.c: In function ‘main’:
main.c:21:4: warning: missing sentinel in function call [-Wformat=]
   21 |    i = execl(argv[1], argv[2], 0);
      |    ^
main.c:27:4: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   27 |    wait(NULL);
      |    ^~~~

Insufficient arguments to load program
Usage: ./a.out <path> <cmd>


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Menampilkan status file menggunakan perintah system call ‘stat’
   Membuat program dengan algoritma sebagai berikut
   (contoh code ada di bagian berikutnya):
   a. Gunakan ‘nama file’ yang diberikan melalui argumen dalam perintah command- line.
   b. Jika ‘nama-file’ tidak ada maka stop disini (keluar program)
   c. Panggil system call ‘stat’ pada ‘nama-file’ tersebut yang akan mengembalikan sebuah struktur.
   d. Tampilkan informasi mengenai st_uid, st_blksize, st_block, st_size, st_nlink, etc.
   e. Ubah waktu dalam st_time, st_mtime dengan menggunakan fungsi ctime.
   f. Bandingkan st_mode dengan konstanta mode seperti S_IRUSR, S_IWGRP, S_IXOTH dan tampilkan informasi mengenai ‘file-permissions’.
   g. Stop
   h. Kode Program & Output:

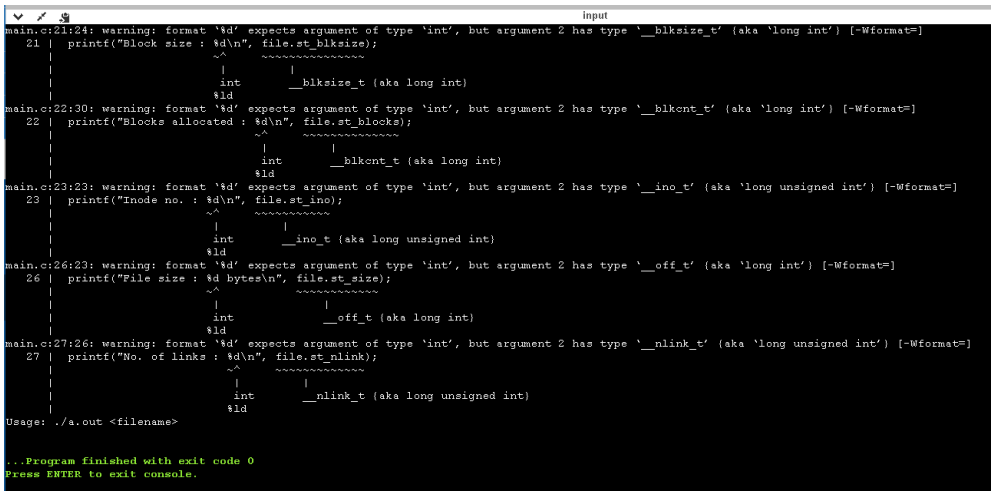
```c
#include <stdio.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <time.h>

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("Blocks 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 : %d bytes\n", file.st_size);
    printf("No. of links : %d\n", file.st_nlink);
    printf("Permissions : ");
    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_IRGRP) ? "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");
    if(file.st_mode & S_IFREG)
        printf("File type: Regular\n");
    if(file.st_mode & S_IFDIR)
        printf("File type: Directory\n");
}

```


input
main.c:21:24: warning: format '%d' expects argument of type 'int', but argument 2 has type '__blksize_t' (aka 'long int') [-Wformat=]
21 | printf("Block size : %d\n", file.st_blksize);
| ~^ ~~~~~~~~~~~~~~~
| | |
| int __blksize_t (aka long int)
| %ld
main.c:22:30: warning: format '%d' expects argument of type 'int', but argument 2 has type '__blkcnt_t' (aka 'long int') [-Wformat=]
22 | printf("Blocks allocated : %d\n", file.st_blocks);
| ~^ ~~~~~~~~~~~~~~~
| | |
| int __blkcnt_t (aka long int)
| %ld
main.c:23:23: warning: format '%d' expects argument of type 'int', but argument 2 has type '__ino_t' (aka 'long unsigned int') [-Wformat=]
23 | printf("Inode no. : %d\n", file.st_ino);
| ~^ ~~~~~~~~~~
| | |
| int __ino_t (aka long unsigned int)
| %ld
main.c:26:23: warning: format '%d' expects argument of type 'int', but argument 2 has type '__off_t' (aka 'long int') [-Wformat=]
26 | printf("File size : %d bytes\n", file.st_size);
| ~^ ~~~~~~~~~~~
| | |
| int __off_t (aka long int)
| %ld
main.c:27:26: warning: format '%d' expects argument of type 'int', but argument 2 has type '__nlink_t' (aka 'long unsigned int') [-Wformat=]
27 | printf("No. of links : %d\n", file.st_nlink);
| ~^ ~~~~~~~~~~~~
| | |
| int __nlink_t (aka long unsigned int)
| %ld
Usage: ./a.out <filename>
...Program finished with exit code 0
Press ENTER to exit console.

5. Menampilkan isi direktori menggunakan perintah system call ‘readdir’
   Membuat program dengan algoritma sebagai berikut
   (contoh code ada di bagian berikutnya):
   a. Gunakan ‘nama-direktori’ yang diberikan sebagai argumen pada command-line.
   b. Jika direktori tidak ditemukan stop, keluar program
   c. Buka direktori menggunakan perintah system call ‘opendir’ yang akan menghasilkan sebuah struktur.
   d. Baca direktori menggunakan perintah system call ‘readdir’ yang juga akan menghasilkan struktur data.
   e. Tampilkan d_name (nama direkrori)
   f. Akhiri pembacaa direktori dengan perintah system call ‘closedir’.
   g. Stop
   h. Kode Program & Output:

main.c

```c
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>

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);
}

```

inp

```
    5 | main(int argc, char *argv[]){
      | ^~~~
Usage: ./a.out <dirname>


...Program finished with exit code 0
Press ENTER to exit console.
```