PRAKTIKUM SISTEM OPERASI

MODUL 8

System Call

Disusun oleh :

Risyma Muti' S.A

L200210228

PROGRAM STUDI TEKNIK INFROMATIKA

FAKULTAS KOMUNIKASI DAN

INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA TAHUN

2022/2023

Langkah 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’, tampilkan ID 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 shellmenggunakan perintah system call getppid.
   f. Stop

Kode program :

```
 GNU nano 6.2                                                    fork.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

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

        if (pid < 0)
        {
                printf("Process creation error"); exit(-1);
        }

        else if (pid == 0)
        {
                printf("Child 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("Parent process:");
                printf("\nProcess id is %d", getpid());
                printf("\nValue of x is %d", x);
                printf("\nProcess id of shell  is %d\n", getppid());
        }
}
```

Output :

```
izam@izam-VirtualBox:~$ cd c
bash: cd: c: No such file or directory
izam@izam-VirtualBox:~$ ls
a.out  Desktop  dirlist.c  Documents  Downloads  exec.c  fork.c  fork.c.save  l200210144.c  Music  Pictures  Public  snap  stat.c  Templates  Videos  wait.c
izam@izam-VirtualBox:~$ gcc fork.c
izam@izam-VirtualBox:~$ ./a.out
Parent process:
Process id is 6109
Value of x is 6
Process id of shell  is 6084
Child process:
Process id is 6110
Value of x is 6
Process id of parent is 1642

izam@izam-VirtualBox:~$
```

2. Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunkaan perintahsystem 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 programdengan 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/d10, selanjutnya tampilkan teks ‘child ends’
   e. Stop

   Kode progam :

```
 GNU nano 6.2                                                    wait.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
        pid_t pid;
        int i, status;
        pid = fork();

        if (pid < 0)
        {
                printf("\nPembuatan proses gagal\n");
                exit(-1);
        }

        else if (pid > 0)
        {
                printf("\nParent starts\nNomor Genap:");
                for (i=2;i<=10;i+=2)
                {
                        printf("%3d", i);
                }
                printf("\nParent ends\n");
        }

        else if (pid == 0)
        {
                printf("\nChild starts\nNomor Ganjil:");
                for (i=1;i<=10;i+=2)
                {
                        printf("%3d", i);
                }
                printf("\nChild ends\n");
        }
}
```

   Output :

```
izam@izam-VirtualBox:~$ ls
a.out      Documents  fork.c        Music     snap       Videos
Desktop    Downloads  fork.c.save   Pictures  stat.c     wait.c
dirlist.c  exe.c      l200210144.c  Public    Templates
izam@izam-VirtualBox:~$ gcc wait.c
izam@izam-VirtualBox:~$ ./a.out

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

Child starts
Nomor Ganjil:  1  3  5  7  9
Child ends
izam@izam-VirtualBox:~$
```

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-lineberhenti (stop).
   b. Membuat child proses dengan perintah system call ‘fork’
   c. Jika return value adalah -1, selanjuntya tampilkan teks ‘Pembuatan proses Gagal’, dan keluar programdengan 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 lokasiyang 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

Kode progam :

```
 GNU nano 6.2                                                    exe.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.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");
                exit(-1);
        }

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

Output :

```
izan@izan-VirtualBox:~$ nano exe.c
izan@izan-VirtualBox:~$ nano exe.c
izan@izan-VirtualBox:~$ ls
a.out  Desktop  dirlist.c  Documents  Downloads  exe.c  fork.c  fork.c.save  l200210144.c  Music  Pictures  Public  snap  stat.c  Templates  Videos  wait.c
izan@izan-VirtualBox:~$ gcc exe.c
izan@izan-VirtualBox:~$ ./a.out bin/ls ls
Child Terminated
Child process
izan@izan-VirtualBox:~$ ls program not loaded using exec system call
```

4. Menampilkan status file menggunakan perintah systemcall ‘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.

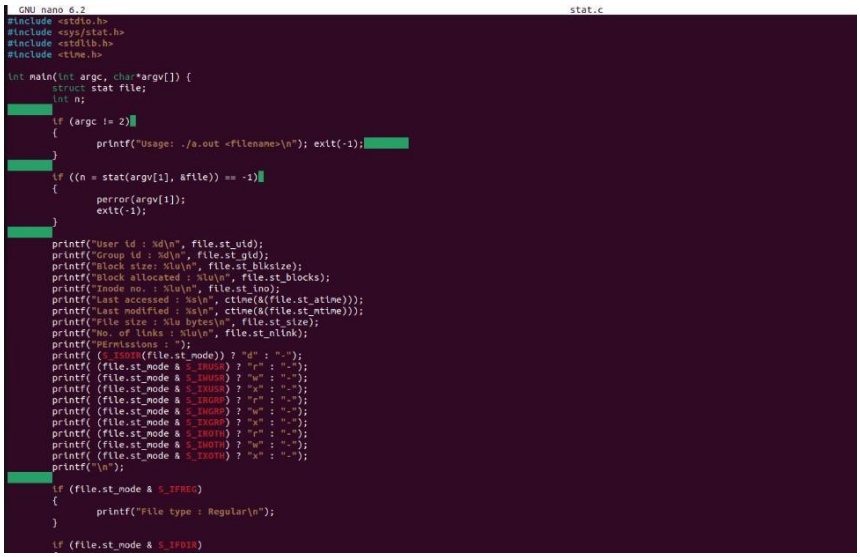
d. Tampilkan informasi mengenai st_uid, st_blksize,st_block, st_size, st_nlink, etc.

e. Ubah waktu dalam st_time, st_mtime dengan menggunakan fungsi ctime.

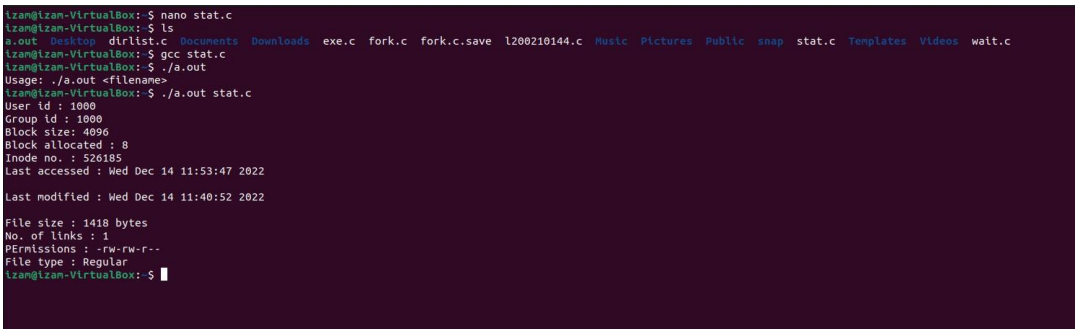
f. Bandingkan st_mode dengan konstanta mode seperti S_IRUSR, S_IWGRP, S_IXOTH dan tampilkan informasi mengenai ‘file-permissions’.

g. Stop

Kode progam :



Output :



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

Kode progam :

```
 GNU nano 6.2                                                    dirlist.c
#include <stdio.h>
#include <dirent.h>
#include <stdlib.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);

        return 0;
}
```

Output :

```
izan@izan-VirtualBox:~$ nano dirlist.c
izan@izan-VirtualBox:~$ ls
a.out  Desktop  dirlist.c  Documents  Downloads  exe.c  fork.c  fork.c.save  l200210144.c  Music  Pictures  Public  snap  stat.c  Templates  Videos  wait.c
izan@izan-VirtualBox:~$ gcc dirlist.c
izan@izan-VirtualBox:~$ ./a.out
Usage: ./a.out <dirname>
izan@izan-VirtualBox:~$ ./a.out /home/izan/t.
/home/izan/t.: No such file or directory
izan@izan-VirtualBox:~$ ./a.out dirlist.c
dirlist.c: Not a directory
izan@izan-VirtualBox:~$ ./a.out /home/izan
Templates
Pictures
.bash_history
snap
fork.c
dirlist.c
.ssh
.
.sudo_as_admin_successful
.cache
.gnupg
.bash_logout
exe.c
.bashrc
..
l200210144.c
Public
Desktop
Music
a.out
.profile
.config
fork.c.save
wait.c
Documents
stat.c
.local
Downloads
Videos
izan@izan-VirtualBox:~$
```