AMELIA RISANTI H
L200210222

PRAKTIKUM SISTEEM OPRASI MODUL 8

KEGIATAN PRAKTIKUM:

1. Buat 'proses anak' (proses baru) menggunakan system call 'fork'. Buat program dengan algoritma berikut: (contoh program diberikan di bagian selanjutnya). sebuah. Deklarasi variabel x yang akan diakses secara bersama antara proses anak dan proses induk.
2. Buat proses anak menggunakan fork system call.
3. Jika nilai pengembalian adalah -1, tampilkan teks 'Membuat proses GAGAL', diikuti dengan keluar dari program dengan perintah panggilan sistem 'keluar'.
4. Jika nilai pengembalian sama dengan 0 (NOL), tampilkan teks 'Child Process', tampilkan ID proses dari proses anak menggunakan perintah panggilan sistem 'getpid', tampilkan nilai x, dan tampilkan ID proses induk dengan perintah panggilan sistem 'getppid'.
5. Untuk nilai pengembalian lainnya, tampilkan teks 'Proses induk', tampilkan ID proses induk menggunakan perintah system call getpid, tampilkan nilai x, dan tampilkan ID proses shell menggunakan perintah system call getppid.
6. Berhenti
7. 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("\n¥alue 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("\n¥alue 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.
```

1. Jeda (blokir) proses induk hingga proses anak selesai, menggunakan perintah panggilan sistem 'tunggu'. Buat program dengan algoritma berikut, contoh program diberikan di bagian selanjutnya. sebuah. Buat proses anak menggunakan system call 'fork'.
2. Jika nilai pengembalian adalah -1, maka tampilkan teks 'membuat proses gagal', dan keluar dari program menggunakan perintah panggilan sistem 'keluar'.
3. Jika return value adalah angka positif (>0), 'pause' menghentikan sementara proses 'parent', menunggu hingga proses anak berakhir dengan menggunakan perintah system call 'wait'. Tampilkan teks 'Parent start', lalu tampilkan angka genap mulai dari 0 sampai 10, terakhir tampilkan teks 'Parent end'.
4. Jika return value adalah 0 (ZERO), tampilkan teks 'Child start', tampilkan angka ganjil mulai dari 0 sampai 10, lalu tampilkan teks 'child ends'
5. Berhenti
6. 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.
```

1. Memuat program yang dapat dieksekusi dalam proses anak menggunakan panggilan sistem exec. Buat program dengan algoritma berikut: (contoh program diberikan di bagian selanjutnya). sebuah. Jika ada 3 argumen di command-line stop (stop).
2. Buat proses anak dengan perintah panggilan sistem 'fork'
3. Jika nilai pengembaliannya adalah -1, maka tampilkan teks 'Membuat proses Gagal', dan keluar dari program dengan perintah system call exit.
4. Jika nilai pengembalian > 0 (positif), maka hentikan proses induk untuk sementara hingga proses anak berakhir menggunakan perintah system call wait. Tampilkan teks 'Anak berakhir', dan hentikan proses induk.
5. Jika nilai kembalian sama dengan 0 (NOL), maka tampilkan teks 'Anak dimulai', muat program dari lokasi yang diberikan di 'jalur' ke dalam proses anak, menggunakan perintah panggilan sistem 'exec'. Jika nilai kembalian dari perintah 'exec' adalah angka negatif, tampilkan kesalahan yang terjadi dan hentikan. Hentikan proses anak.
6. Berhenti
7. Outpt :

```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.
```

1. Menampilkan status file menggunakan perintah system call 'stat' Buat program dengan algoritma berikut (contoh kode ada di bagian selanjutnya): Gunakan 'nama file' yang disediakan melalui argumen di baris perintah.
2. Jika 'nama file' tidak ada maka berhenti di sini (keluar dari program)
3. Panggil panggilan sistem 'stat' pada 'nama file' yang akan mengembalikan struktur.
4. Menampilkan informasi tentang st_uid, st_blksize, st_block, st_size, st_nlink, dll. e.Ubah waktu di st_time, st_mtime dengan menggunakan fungsi ctime.
5. Bandingkan st_mode dengan konstanta mode seperti S_IRUSR, S_IWGRP, S_IXOTH dan tampilkan informasi tentang 'izin file'.
6. Berhenti
7. 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>
```

1. Tampilkan isi direktori menggunakan perintah system call 'readdir' Buat program dengan algoritma berikut (contoh kode ada di bagian selanjutnya):
2. Gunakan 'nama direktori' yang diberikan sebagai argumen pada baris perintah. b.Jika direktori tidak ditemukan berhenti, keluar dari program
3. Buka direktori menggunakan perintah panggilan sistem 'opendir' yang akan menghasilkan struktur.di.Baca direktori menggunakan perintah system call 'readdir' yang juga akan menghasilkan struktur data.
4. Tampilkan d_name (nama direktori)
5. Akhiri pembacaan direktori dengan perintah system call 'closedir'.
6. Berhenti
7. 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.
```