**A. Simulator Sistem Operasi**

Nama Berkas

OS.c

Deskripsi

Simulator Sistem Operasi berisi fungsi alokasi dan dealokasi memori virtual berupa *shared memory* yang akan digunakan oleh Unit Manajemen Memori (MMU). Selain alokasi dan dealokasi, proses pemuatan(*load*) page yang diminta (*request*) oleh MMU juga dilakukan pada Simulator Sistem Operasi.

Tujuan

Sebagai Sistem Operasi bagi MMU dalam melakukan alokasi, dealokasi memori virtual, penanganan *page* baik pemuatan/*load* *page* dan cara memilih *page*

Pseudocode

*Masukan* page\_table\_entry PageTable[]

int NumberOfPages

*Keluaran* -

*Variabel* int banyakPage

int banyakFrame

page\_table\_pointer dataPage

int statusOS ← 0

int aksesDisk ← 0

*Keterangan* terdapat int argc dan char \*argv[] sebagai parameter awal program

*Algoritma*

if argc < 3 then

cetak("> Kesalahan: argumen Page(arg1) dan Frame(arg2) tidak boleh kosong")

cetak("> Gunakan perintah:")

cetak(" “, argv0, ” <banyak page> <banyak frame>")

cetak(" contoh: ")

cetak(argv0, “5 2")

→ 0

banyakPage ← atoi(argv1)

banyakFrame ← atoi(argv2)

idSharedMemory ← alokasiSharedMemory(getpid(), banyakPage \*

sizeof(page\_table\_entry))

if idSharedMemory = -1 then

cetak("Alokasi Shared Memory Gagal")

cetak("Program Berhenti.")

→ 0

signal(SIGINT, penangananCtrlC)

signal(SIGUSR1, penangananSIGUSR1)

while statusOS <> -1

pause()

cetak("The MMU has finished")

PrintPageTable(dataPage, banyakPage)

cetak(aksesDisk, " disk accesses required\n\n")

dealokasiSharedMemory(idSharedMemory)

→ 0

Fungsi / Prosedur

**1. Print Page Table**

**Nama**

Print Page Table - Mencetak data pada memori virtual ke layar

**Sinopsis**

#include “PageTable.h”

void PrintPageTable(page\_table\_entry PageTable[], int NumberOfPages);

**Deskripsi**

Prosedur ini menggunakan tipe bentukan page\_table\_entry yang ada pada *header* PageTable.h. Parameter pertama prosedur ini merupakan sebuah pointer ke memori virtual yang dialokasi oleh OS Simulator. NumberOfPages adalah banyaknya *Page* yang dialokasikan oleh OS di awal pada memori virtual PageTable.

Pada pencetakan layar, prosedur ini mencetak anggota(member) tipe bentukan page\_table\_entry pada *Page* yang terdapat pada memori virtual.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* PrintPageTable

*Masukan* page\_table\_entry PageTable[]

int NumberOfPages

*Keluaran* -

*Variabel* int Index

*Algoritma*

Iterasi Index ← 0 ke NumberOfPages - 1

cetak(“ “, Index, “:”,

“ Valid = “, PageTableIndex.Valid,

“ Frame = “, PageTableIndex.Frame,

“ Dirty = “, PageTableIndex.Dirty,

“ Requested = “, PageTableIndex.Requested)

**Contoh**

Masukan

PrintPageTable(dataPage, banyakPage);

Keluaran

0: Valid=0 Frame=-1 Dirty=1 Requested=0

1: Valid=0 Frame=-1 Dirty=0 Requested=0

**Catatan**

Tidak ada

**2. Alokasi Shared Memory**

**Nama**

Alokasi Shared Memory - Melakukan alokasi memori virtual ke disk

**Sinopsis**

#include <sys/ipc.h>

#include <sys/shm.h>

#include <errno.h>

#include “PageTable.h”

int alokasiSharedMemory(key\_t kunci, int ukuran);

**Deskripsi**

Fungsi ini melakukan alokasi memori virtual ke disk (bukan RAM) sebagai *shared memory* dengan *key identifier* (*Shared Memory Key*) [baca: shmget] adalah proses PID program OS ini. Penggunaan PID Proses sebagai kunci identifikasi (*Shared Memory Key*) agar MMU dapat menunjuk PID program dan *Shared Memory Key* dengan angka yang sama.

Alokasi tipe data page\_table\_entry pada memori virtual diinisiasi dengan Valid = 0, Frame = -1, Dirty = 0, dan Requested = 0.

**Nilai Kembali**

Jika alokasi berhasil, mengembalikan *Shared Memory ID*(int), mengembalikan -1 jika alokasi gagal.

**Pseudocode**

*Prosedur* alokasiSharedMemory

*Masukan* key\_t kunci

int ukuran

*Keluaran* int shmid {Shared Memory ID}

*Variabel* int i

page\_table\_pointer dataPage {GLOBAL}

*Algoritma*

If (shmid ← shmget(kunci, ukuran, 0644 | IPC\_CREAT)) = -1 then

perror("shmget")

→ -1

dataPage ← (page\_table\_pointer) shmat(shmid, NULL, 0)

If dataPage = (page\_table\_pointer)(-1) then

perror("shmat")

→ -1

cetak("The shared memory key (PID) is ", kunci)

cetak("Initialized page table")

iterasi i ← 0 ke banyakPage - 1

dataPagei.Valid ← 0

dataPagei.Frame ← -1

dataPagei.Dirty ← 0

dataPagei.Requested ← 0

→ shmid

**Contoh**

Masukan

int idSharedMemory ← alokasiSharedMemory(getpid(),

banyakPage\*sizeof(page\_table\_entry))

Keluaran

Jika alokasi berhasil idSharedMemory bernilai >= 0, bernilai -1 jika gagal

**Catatan**

Tidak ada

**3. Dealokasi Shared Memory**

**Nama**

Dealokasi *Shared Memory* - Melakukan dealokasi memori virtual dari disk

**Sinopsis**

#include <sys/ipc.h>

#include <sys/shm.h>

#include <errno.h>

#include “PageTable.h”

int dealokasiSharedMemory(int kunci);

**Deskripsi**

Dealokasi memori virtual yang telah dialokasi dapat dilakukan setelah memori virtual yang ditempatkan (*attach*) pada RAM dilepas (*detach*). Setelah proses *detach shared memory*, barulah proses dealokasi dilakukan. Ketika proses dealokasi tidak dilakukan, maka memori virtual yang dialokasi sebelumnya akan berada pada disk selamanya. *Shared memory* yang telah dilepas (*detach*) dari RAM dapat ditempatkan (*attach*) kembali ke RAM. Setelah proses dealokasi memori virtual akan terhapus dan tidak dapat dikembalikan lagi.

**Nilai Kembali**

Jika dealokasi berhasil, mengembalikan 0, mengembalikan -1 jika alokasi gagal.

**Pseudocode**

*Prosedur* dealokasiSharedMemory

*Masukan* int kunci

*Keluaran* int

*Variabel* page\_table\_pointer dataPage {GLOBAL}

*Algoritma*

if shmdt(dataPage) = -1 then

perror("shmdt")

→ -1

if shmctl(kunci, IPC\_RMID, NULL) = -1 then

perror("shmctl")

→ -1

→ 0

**Contoh**

Masukan

If dealokasiSharedMemory(idSharedMemory) = -1 then

cetak("Dealokasi Shared Memory Gagal")

else

cetak("Dealokasi Shared Memory Berhasil")

Keluaran

Jika dealokasi berhasil tercetak Dealokasi Shared Memory Berhasil, tercetak Dealokasi Shared Memory Gagal jika gagal.

**Catatan**

Tidak ada

**4. Cari Request**

**Nama**

Cari Request - Mencari *request Page* yang dikirim oleh MMU

**Sinopsis**

#include “PageTable.h”

int cariRequest();

**Deskripsi**

Program MMU melakukan *request Page* yang tidak ada pada *shared memory* dengan cara menandai *Page* yang direquest lalu mengirim sinyal SIGUSR1 ke program OS, kemudia program OS memindai *Page* yang ditandai oleh MMU.

Jika MMU mengirim sinyal SIGUSR1 tanpa melakukan *request* maka fungsi ini akan mengembalikan -1. Nilai kembali -1 menunjukkan MMU telah selesai memproses seluruh masukan, sehingga setelah nilai kembali -1, Simulator Sistem Operasi akan berhenti.

**Nilai Kembali**

Jika request ditemukan, mengembalikan indeks *Page* yang ditandai oleh MMU, mengembalikan -1 jika *request* tidak ditemukan.

**Pseudocode**

*Prosedur* cariRequest

*Masukan* -

*Keluaran* int

*Variabel* int i

int banyakPage {GLOBAL}

*Algoritma*

iterasi i ← 0 ke banyakPage - 1

Jika dataPagei.Requested <> 0 then

→ i

→ -1

**Contoh**

Masukan

int i ← cariRequest()

Keluaran

Jika MMU menandai sebuah *Page* pada memori virtual sebagai bentuk *request*, i bernilai indeks *Page* yang ditandai oleh MMU, namun mengembalikan -1 jika MMU tidak melakukan *request*

**Catatan**

Fungsi ini digunakan oleh prosedur void tanganiData(int nomor\_request)

**5. Penanganan Sinyal SIGUSR1**

**Nama**

Penanganan Sinyal SIGUSR1 - Memberikan respon terhadap sinyal SIGUSR1 yang dikirim oleh program MMU.

**Sinopsis**

#include <signal.h>

#include <errno.h>

#include “PageTable.h”

void penangananSIGUSR1();

**Deskripsi**

Program MMU akan mengirimkan sinyal SIGUSR1 ke Simulator Sistem Operasi dalam dua kejadian, pertama ketika ingin melakukan *request Page* yang tidak ada pada *shared memory*, kedua pada saat seluruh masukan telah selesai dikerjakan oleh MMU dan ingin memberitahu OS bahwa pekerjaan telah selesai.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* penangananSIGUSR1

*Masukan* -

*Keluaran* -

*Variabel* int no\_request

int statusOS {GLOBAL}

Int PIDMMU

*Algoritma*

no\_request ← cariRequest()

statusOS ← no\_request

if no\_request <> -1 then

PIDMMU ← dataPageno\_request.Requested

tanganiData(no\_request)

cetak("Unblock MMU")

if kill(PIDMMU, SIGCONT) == -1 then

perror("Gagal mengirim sinyal SIGCONT ke MMU")

statusOS ← -1

signal(SIGUSR1, penangananSIGUSR1);

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**6. Penanganan Interup Keyboard**

**Nama**

Penanganan *Interup Keyboard* - Menghentikan Sistem Operasi ketika sedang berjalan melalui *keyboard* dengan Ctrl-C.

**Sinopsis**

#include <signal.h>

void penangananCtrlC(int sig)

**Deskripsi**

Program Simulator Sistem Operasi dapat dihentikan ketika sedang berjalan menggunakan Ctrl-C. Prosedur ini menggunakan sinyal SIGINT yang dikirim jika ada *interrupt* melalui *keyboard*.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* penangananCtrlC

*Masukan* int sig

*Keluaran* -

*Variabel* char c

int statusOS {GLOBAL}

*Algoritma*

signal(sig, SIG\_IGN);

cetak("Anda menekan Ctrl-C? Ingin Berhenti? [y/t] : ")

c ← getchar()

if c = 'y' OR c = 'Y' then

statusOS ← -1

else

signal(SIGINT, penangananCtrlC)

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**7. Penanganan Data**

void tanganiData(int nomor\_request)