Dibuat oleh :

Febi Agil Ifdillah (13514010)

Harry Octavianus Purba,

Rio Chandra Rajaguguk

**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

int frameVictim

Vector vectorFrame

*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

dataPagei.Counter ← 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. Cek Frame – Penuh**

**Nama**

Cek Frame – Penuh

**Sinopsis**

#include “Boolean.h”

#include “PageTable.h”

boolean isFrameFull()

**Deskripsi**

Melakukan pengecekan apakah kondisi frame sekarang penuh atau tidak. Jika penuh akan mengembalikan nilai true, dan false jika sebaliknya.

**Nilai kembali**

boolean (true atau false)

**Pseudocode**

*Fungsi* isFrameFull()

*Masukan -*

*Keluaran* boolean(true atau false)

*Variable* int i

int hitung ← 0

int temp

int banyakFrame {GLOBAL}

Vector vectorFrame {GLOBAL}

*Algoritma*

iterasi i←0 ke banyakFrame-1

temp ← vector\_get(vectorFrame, i)

if (temp !=-1)

hitung=hitung+1

if (hitung=banyakFrame)

→ true

else

→ false

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**8. Cek Frame – Kosong**

**Nama**

Cek Frame – Kosong

**Sinopsis**

#include “Boolean.h”

#include “PageTable.h”

boolean isFrameEmpty()

**Deskripsi**

Melakukan pengecekan apakah kondisi frame sekarang kosong atau tidak. Jika penuh akan mengembalikan nilai true, dan false jika sebaliknya.

**Nilai kembali**

boolean (true atau false)

**Pseudocode**

*Fungsi* isFrameEmpty()

*Masukan -*

*Keluaran* boolean(true atau false)

*Variable* int i

int hitung ← 0

int temp

int banyakFrame {GLOBAL}

Vector vectorFrame {GLOBAL}

*Algoritma*

iterasi i←0 ke banyakFrame-1

temp ← vector\_get(vectorFrame, i)

if (temp !=-1)

hitung=hitung+1

if (hitung=0)

→ true

else

→ false

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**9. Penanganan Data**

**Nama**

Penanganan Data – Menangani data atau page masukan pada nomor request yang sedang berlangsung yang kemudian akan diproses

**Sinopsis**

#include “PageTable.h”

void tanganiData(int nomor\_request)

**Deskripsi**

Page Table dengan nomor request tersebut akan diproses di prosedur tanganiData. Proses akan berbeda sesuai dengan data pada Page Table yang sedang ditangani. Misalnya, bergantung kepada valid atau tidaknya page, juga penuh atau tidaknya frame.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* tanganiData

*Masukan* int nomor\_request

*Keluaran* -

*Variable* int i

int temp

page\_table\_pointer dataPage {GLOBAL}

*Algoritma*

cetak("Process, “ , dataPagenomor\_request ,“has requested page ", nomor\_request)

iterasi i ← 0 ke banyakPage-1

int temp ← dataPagei.Counter

dataPagei.Counter ← temp +1

addToFrame(nomor\_request)

dataPagenomor\_request.Requested ← 0

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**11. Penghapusan data dari frame**

**Nama**

Penghapusan data dari frame – Melakukan penghapusan data dari frame, terjadi karena frame penuh.

**Sinopsis**

#include “Boolean.h”

#include “PageTable.h”

#include “Vector.h”

void addToFrame(int nomor\_page)

**Deskripsi**

Jika *frame* penuh, dan ada data yang harus dimasukkan ke dalam *frame*, maka prosedur ini dipanggil. Salah satu *page* yang ada di *frame* akan menjadi victim dan dihapus dari frame, agar setelah penghapusan tersebut ada sebuah ruang pada *frame* yang akan menjadi tempat untuk page yang sedang melakukan *request* pada waktu pemanggial prosedur ini.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* delFromFrame

*Masukan* int nomor\_page

*Keluaran* -

*Variable* int i

int simpan

int max ← 0

boolean penuh

int banyakPage {GLOBAL}

page\_table\_pointer dataPage {GLOBAL}

int frameVictim {GLOBAL}

Vector vectorFrame {GLOBAL}

*Algoritma*

if (isFrameEmpty())

exit

iterasi i←0 ke banyakPage-1

if ((dataPagei.Counter>max) and (dataPagei.Frame != -1))

simpan ← i

max ← dataPagei.Counter

break;

cetak("Choose a victim page ", simpan)

cetak("Victim is dirty, write out")

cetak("Put in victim's frame ", dataPage[simpan].Frame)

frameVictim ← dataPagesimpan.Frame

vector\_set(&vectorFrame, frameVictim, 0)

dataPagesimpan.Valid ← 0

//victim di invalid, menyatakan bahwa sudah tidak ada di memory fisik

dataPagesimpan.Frame ← -1

dataPagesimpan.Counter ← 0

if (dataPagesimpan.Dirty = true)

dataPage[simpan].Dirty ← 0

aksesDisk++

sleep(1); //menuliskan ke disk

**Contoh**

Tidak ada

**Catatan**

Akses disk dua kali, pertama untuk menghapus dan kedua untuk menambahkan ke frame.

**12. Penambahan ke frame**

**Nama**

Penambahan ke frame – Melakukan penambahan page ke frame jikadengan beberapa syarat.

**Sinopsis**

#include “Boolean.h”

#include “PageTable.h”

#include “Vector.h”

void addToFrame(int nomor\_page)

**Deskripsi**

Data dengan nomor page yang menjadi parameter dari prosedur ini akan ditangani sebagai berikut :

1. Jika valid, maka tidak akan mengakses disk

2. Jika tidak valid, maka akan dilakukan penambahan dengan syarat :

a. Bila masih ada frame kosong, tempatkan di frame tersebut

b. Jika tidak ada, lakukan proses pencarian victim, yang kemudian akan diswap dengan data page yang sedang ditangani.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* addToFrame

*Masukan* int nomor\_page

*Keluaran* -

*Variable* int i

int temp

boolean penuh

page\_table\_pointer dataPage {GLOBAL}

int frameVictim {GLOBAL}

Vector vectorFrame {GLOBAL}

*Algoritma*

if (!dataPageno\_page.Valid)

penuh← false

if (isFrameFull())

delFromFrame()

penuh ← true

dataPageno\_page.Valid ← 1

if (penuh)

dataPageno\_page.Frame ← frameVictim

vector\_set(vectorFrame, frameVictim, no\_page)

else

//Jika belum ada di frame, cari frame yang belum ditempati

//Jika terdapat frame yang tidak ditempati, alokasikan.

Iterasi i← 0 ke i<banyakFrame-1

temp ← vector\_get(vectorFrame, i)

if (temp = -1) {

dataPageno\_page.Frame ← i

vector\_set(vectorFrame, i, no\_page)

cetak("Put it in free frame ", i)

break

aksesDisk++

sleep(1) //simulasi akses disk

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**B. Array Dinamis : Vector**

Nama Berkas

Vector.c

Deskripsi

Vector ini berisi semua fungsi yang dibutuhkan sebagai container, mampu menyimpan banyak elemen ukurannya dapat berubah secara dinamis, bergantung kebutuhan.

Tujuan

Sebagai sebuah dynamic array yang menjadi representasi dari frame yang digunakan untuk menyimpan page.

Pseudocode

*Masukan* -

*Keluaran*  -

*Variabel* type Vector = <

size : integer

capacity : integer

data : address

>

*Keterangan* Algoritma di bawah, hanyalah implementasi dari vector.h

*Algoritma*

prosedur vector\_init(vector : address)

prosedur vector\_append(vector : address ,value:integer)

fungsi vector\_get(vector: address, index:integer) → integer

prosedur vector\_set(vector :address, index: integer, value: integer)

prosedur vector\_double\_capacity(vector:address)

prosedur vector\_free(vector: address)

Fungsi/Prosedur

**1. Inisialisasi Vector**

**Nama**

Inisialisasi Vector

**Sinopsis**

#include “Vector.h”

void vector\_init(Vector \* vector)

**Deksripsi**

Vector akan diinisialisasi dengan ukuran 0, kapasitas awal 50, berisi data kosong.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* vector\_init

*Masukan* addressVector pointer vector {GLOBAL}

*Keluaran -*

*Variable -*

*Algoritma*

vector↑.size ← 0

vector↑.capacity ← 50

vector↑. data = memoryallocation(size(int) \* vector↑. capacity

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**2. Append Vector**

**Nama**

Append Vector

**Sinopsis**

#include “Vector.h”

void vector\_append(Vector \* vector, int value)

**Deksripsi**

Apabila akan menginsert element namun ukuran dari vector melebihi kapasitas, maka ia akan diperbesar ukurannya dengan prosedur ini. Setelah diperbesar dan cukup, maka kita dapat melakukan insert pada indeks tersebut.

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur vector\_append*

*Masukan*

address vector

int value

*Keluaran -*

*Variable -*

*Algoritma*

vector\_double\_capacity(vector)

vector↑. datavector->size++ ← value

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**3. Get Vector**

**Nama**

Get Vector

**Sinopsis**

#include “Vector.h”

int vector\_get(Vector \* vector, int index)

**Deksripsi**

Mendapatkan elemen pada index ke – index, yang ada di vector.

**Nilai Kembali**

integer

**Pseudocode**

*Prosedur vector\_get*

*Masukan*

address vector

int index

*Keluaran* int

*Variable -*

*Algoritma*

if (index >= vector↑.size or index < 0)

cetak(“Index ”, index, “ out of bounds for vector of size “,

vector↑.capacity)

→ vector↑.dataindex

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**4. Set Vector**

**Nama**

Set Vector

**Sinopsis**

#include “Vector.h”

void vector\_set(Vector \* vector, int index, int value)

**Deksripsi**

Melakukan set terhadap index ke-index dengan data value

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur vector\_set*

*Masukan*

address vector

int index

int value

*Keluaran -*

*Variable -*

*Algoritma*

while (index>=vector↑.size)

vector\_append(vector, 0)

vector↑.dataindex ← value

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**5. Menambah ruang Vector**

**Nama**

Menambah ruang vector

**Sinopsis**

#include “Vector.h”

void vector\_double\_capacity(Vector \* vector)

**Deksripsi**

Membuat ruang vector menjadi dua kali lipat sebelumnya

**Nilai Kembali**

Tidak ada

**Pseudocode**

*Prosedur* vector\_double\_capacity

*Masukan* address vector

*Keluaran-*

*Variable -*

*Algoritma*

if (vector↑.size >= vector↑.capacity)

vector↑.capacity \* ← 2

vector↑.data ← realloc(vector↑.data, sizeof(int) \* vector↑.capacity)

**Contoh**

Tidak ada

**Catatan**

Tidak ada

**6. Mengosongkan Vector**

**Nama**

Mengosongkan vector

**Sinopsis**

#include “Vector.h”

void vector\_free(Vector \* vector)

**Deksripsi**

Melakukan pengosongan terhadap vector jika sudah selesai digunakan

**Nilai Kembali**

Tidak ada

**Pseudocode**

free(vector↑.data)

**Contoh**

Tidak ada

**Catatan**

Tidak ada