***Коваленко Сергей***

***Группа ИВ-63***

***Кафедра ВТ***

***Факультет ИВТ***

***Операционные системы***

***Лабораторная работа №2***

***Задание:*** *разработать модель системы со страничной организацией памяти, в которой для управления страницами реализуется алгоритм Aging.*

***Описание структуры****: Система состоит из глобального управляющего класса, принимающего решения о загрузке и замещении страниц, в состав которого входят:*

*массив дескрипторов процессов.*

*массив дескрипторов страниц в физической памяти.*

*массив дескрипторов страниц в своп.*

*Каждый процесс в свою очередь содержит карту используемых страниц памяти.*

*Функциональность процесса:*

*1. запрос страницы.*

*2. вывод информации об используемых страницах памяти.*

*Функциональность управляющего класса:*

*1. загрузка страницы.*

*2. замещение страницы.*

*3. изменение «возраста» вновь вызванной страницы.*

*4. реализация шага моделирования (выбор процесса, требуемая манипуляция со страницами в памяти)*

*5. управления ходом моделирования и выводом отчетов.*

*Как происходит моделирование:*

*1. инициализация системы, памяти и процессов.*

*2. моделирование происходит пошагово. На каждом шаге выбирается процесс и активизируемая страница виртуальной памяти. Далее управляющий класс System осуществляет принятие решения в зависимости от состояния требуемой страницы.*

*2.1. Если страница уже в физической памяти, все страницы физической памяти помечаются, как устаревшие еще на 1 такт, требуемая страница обозначается как только что запрошенная.*

*2.2. Страница выгружена из физической памяти. В таком случае страница загружается обратно в память, вместо самой старой страницы памяти. Последняя выгружается из памяти. Остальные действия аналогичны предыдущим.*

*2.3. Страница ранее не использовалась. Тогда выбирается свободное пространство в файле подкачки, куда выгружается наиболее старая активная страница. Вместо нее загружается требуемая страница памяти. Дальнейшие действия аналогичны п. 2.1.*

*В требуемых местах вызывается вывод отчета на монитор и в файл, если таковой был определен.*

*Программный код:*

/\*

\* process.h

\*

\* Created on: 04.12.2009

\* Author: Shieghtun

\*/

#ifndef PROCESS\_H\_

#define PROCESS\_H\_

#include <stdio.h>

#include <conio.h>

class process

{

private:

void\*\* pageDescryptors;

public:

int pages;

int activePages;

void init();

void\*\* selectPage();

void stateDump(FILE\* dst);

};

#endif /\* PROCESS\_H\_ \*/

/\*

\* system.h

\*

\* Created on: 05.12.2009

\* Author: Shieghtun

\*/

#ifndef SYSTEM\_H\_

#define SYSTEM\_H\_

#include "process.h"

class pageDescriptor

{

private:

public:

void\*\* page;

unsigned long memUsing;

};

class system

{

private:

static int physPages;

static int swapPages;

static int processAmount;

static pageDescriptor\* operativeMemory;

static pageDescriptor\* swap;

static process\* managed;

static void loadPage(void\*\* page);

static void reloadPage(void\*\* page);

static void callPage(void\*\* page);

static int findPlace();

public:

static void init();

static void run(int steps);

static void printReport(FILE\* dst, int processId);

};

#endif /\* SYSTEM\_H\_ \*/

/\*

\* process.cpp

\*

\* Created on: 05.12.2009

\* Author: Shieghtun

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include "process.h"

void process::init()

{

printf("Configure next process\n");

printf("Input amount of overall pages, used by process\n");

scanf("%d", &pages);

printf("Input amount of active pages");

scanf("%d", &activePages);

pageDescryptors = (void\*\*)calloc(pages, 4);

}

void\*\* process::selectPage()

{

int p;

if (((rand()\*(float)1.0)/RAND\_MAX) < 0.9)

{

p = (int)floor((rand()\*(float)activePages)/RAND\_MAX);

printf("page selected %d\n", p);

return &(pageDescryptors[p]);

}

else

{

p = ((int)floor((rand()\*(float)(pages - activePages))/RAND\_MAX))

+ activePages;

printf("page selected %d\n", p);

return &(pageDescryptors[p]);

}

}

void process::stateDump(FILE\* dst) //fputs don't take list of variables redo all this

{

int i = 0;

char msg[255];

char tmp[20];

for(i = 0; i < pages; i++)

{

// fputs don't take list of variables redo all this

strcpy(msg, "Page ");

strcpy(tmp, "");

\_itoa(i, tmp, 10);

strcat(msg, tmp);

void\*\* t = (void\*\*)((unsigned long)pageDescryptors + 4\*i);

if (\*t != NULL)

{

strcat(msg, "was activated");// may be troubles coz may be nonlinear reflection

if ((((unsigned int)(pageDescryptors[i])) & 0x00000004) == 0)

strcat(msg, " but now swapped.\n");

else

strcat(msg, " and now is in physical memory.\n");

}

else

strcat(msg, "was not activated\n");

printf(msg);

if (dst!=NULL)

{

fprintf(dst, msg);

}

}

}

/\*

\* system.cpp

\*

\* Created on: 05.12.2009

\* Author: Shieghtun

\*/

#include <math.h>

#include <stdlib.h>

#include "system.h"

int system::processAmount;

int system::physPages;

int system::swapPages;

pageDescriptor\* system::operativeMemory;

pageDescriptor\* system::swap;

process\* system::managed;

void system::init()

{

printf("Initialize overall model:\n");

printf("Amount of pages in physical memory of model is 32\n");

physPages = 32;

operativeMemory = new pageDescriptor[32];

int i;

for (i = 0; i < 32; i++)

{

operativeMemory[i].memUsing = 0;

operativeMemory[i].page = NULL;

}

printf("Input amount of processes: \n");

scanf("%d", &processAmount);

managed = new process[processAmount];

swapPages = 0;

for (i = 0; i < processAmount; i++)

{

managed[i].init();

swapPages += managed[i].pages;

}

swapPages -= 32;

if (swapPages > 0)

{

swap = new pageDescriptor[swapPages];

for (i = 0; i < swapPages; i++)

{

swap[i].memUsing = 0;

swap[i].page = NULL;

}

}

else swapPages = 0;

}

void system::printReport(FILE\* dst, int processId)

{

int i = 0;

if (processId != 0)

for (i = 0; i < processAmount; i++)

managed[i].stateDump(dst);

else

managed[processId].stateDump(dst);

}

void system::run(int steps)

{

int processId;

int i;

for (i = 0; i < steps; i++)

{

processId = (int)floor(rand()\*((float)processAmount)/RAND\_MAX);

printf("Process # %d\n", processId);

void\*\* page = managed[processId].selectPage();

if ((((unsigned long)(\*page)) & 0xfffffff8) == NULL)

loadPage(page);

else if((((unsigned long)(\*page))&0x00000004) == 0)

reloadPage(page);

else

callPage(page);

\*page = (void\*)((unsigned long)\*page | 0x00000004);

}

}

void system::callPage(void\*\* page)

{

int i;

for (i = 0; i < 32; i++)

{

operativeMemory[i].memUsing = operativeMemory[i].memUsing >> 1;

}

pageDescriptor\* p = (pageDescriptor\*)((unsigned long)(\*((void\*\*)page)) & 0xfffffff8)

;

(\*p).memUsing = (\*p).memUsing | 0x80000000;

printf("Call used\n");

}

void system::loadPage(void\*\* page)

{

bool loaded = false;

int i;

for (i = 0; ((i < 32) && !loaded); i++)

if (operativeMemory[i].page == NULL)

{

\*page = (void\*)&(operativeMemory[i]);

operativeMemory[i].page = page;

operativeMemory[i].memUsing = 0x80000000;

loaded = true;

}

for(i = 0; ((i < swapPages) && !loaded); i++)

if (swap[i].page == NULL)

{

int place = findPlace();

swap[i].page = operativeMemory[place].page;

\*(operativeMemory[place].page) = (void\*)&(swap[i]);

swap[i].memUsing = 0;

operativeMemory[place].page = page;

operativeMemory[place].memUsing = 0x80000000;

loaded = true;

}

printf("Load page used");

}

void system::reloadPage(void\*\* page)

{

pageDescriptor\* t = (pageDescriptor\*)\*page;

(\*t).page = NULL;

printf("Reload used and ");

loadPage(page);

}

int system::findPlace()

{

int i;

int res = 0;

for(i = 1; i < 32; i++)

if (operativeMemory[i].memUsing < operativeMemory[res].memUsing)

{

operativeMemory[res].memUsing = operativeMemory[i].memUsing;

res = i;

}

return res;

}

/\*

\* main.cpp

\*

\* Created on: 05.12.2009

\* Author: Shieghtun

\*/

#include <conio.h>

#include "process.h"

#include "system.h"

int main()

{

printf("Program model for aging algorithm\n");

printf("Initialize the system:");

system model = system();

model.init();

model.printReport(NULL, 0);

\_getch();

model.run(2000);

model.printReport(NULL, 0);

\_getch();

return 0;

}