Activity 8

ธนัส วงศ์สมุทร 6432067021

ก้องภพ จริยาสถาพร 6430014321

1. FIFO

A screenshot of a computer

Description automatically generated

int *get\_free\_frame*(int page\_number, int timestamp)

{

*if* (num\_free\_frames > 0)

{

// *Get the first free frame*

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].page\_number == -1)

{

// *Assignment 1.1*

// *Update page number and timestamp of the free frame*

frames[i].page\_number = page\_number;

frames[i].timestamp = timestamp;

num\_free\_frames--;

*return* i;

}

}

}

// *If no free frame, select one of occupied frames, using FIFO alorithm.*

*else*

{

int oldest\_frame = 0;

int min\_timestamp;

// *Assignment 1.2*

// *Find the oldest frame that is to be replaced*

min\_timestamp = 1 << 30;

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].timestamp < min\_timestamp)

{

oldest\_frame = i;

min\_timestamp = frames[i].timestamp;

}

}

// *Assignment 1.3*

// *invalidate the replaced page in the page table (valid=0)*

page\_table[frames[oldest\_frame].page\_number].valid = 0;

// *Assignment 1.4*

// *assign page number and timestamp to the selected frame (frames[oldest\_frame])*

frames[oldest\_frame].page\_number = page\_number;

frames[oldest\_frame].timestamp = timestamp;

*return* oldest\_frame;

}

}

1. LRU

*else*

{

// *Assignment 2*

// *Update timestamp of the referenced page in the frames list*

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].page\_number == page\_number)

{

frames[i].timestamp = page\_references;

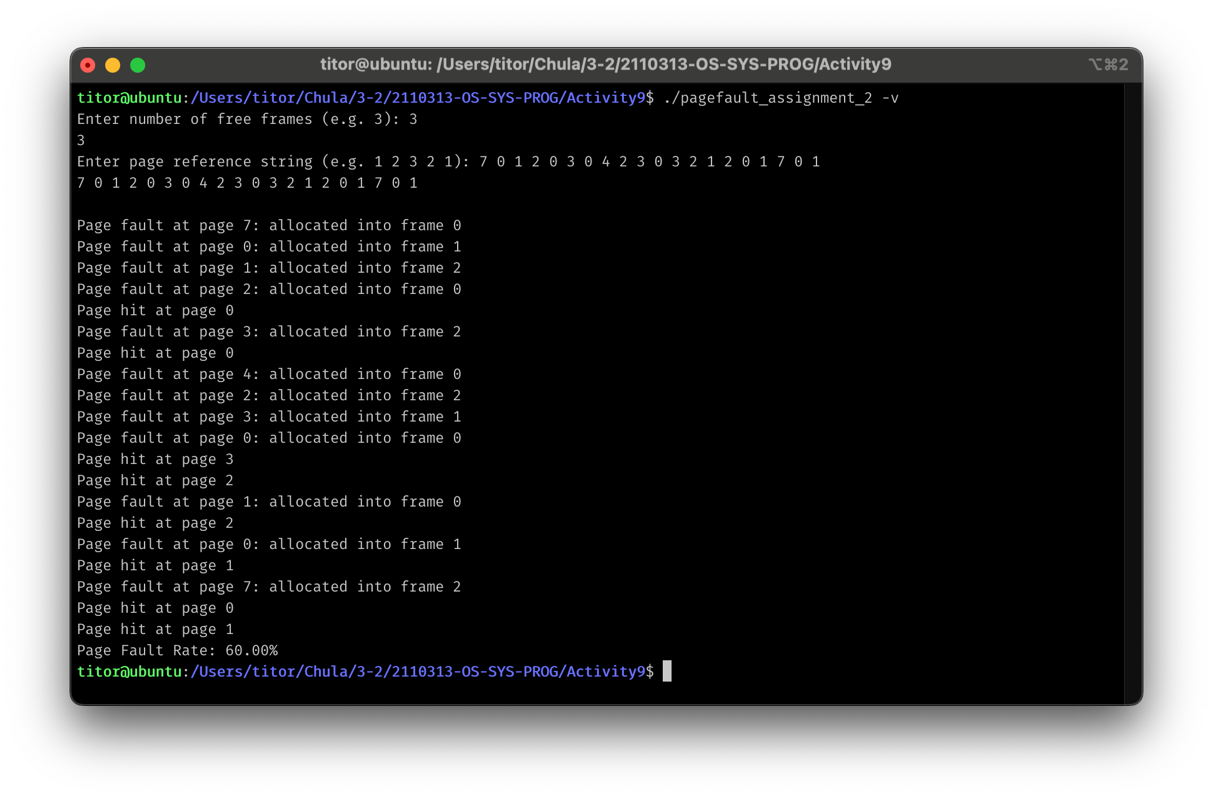
}

}

*if* (verbose)

*printf*("Page hit at page %d\n", page\_number);

}



Full Code

int *main*(int argc, char \*argv*[]*)

{

char buf[5];

int page\_faults = 0, page\_references = 0;

char page\_reference\_string[1024];

int verbose = 0;

// *Parse command line arguments*

*if* (argc > 1 && *strcmp*(argv[1], "-v") == 0)

{

verbose = 1;

}

// *Read in number of free frame*

*printf*("Enter number of free frames (e.g. 3): ");

*fgets*(buf, sizeof(buf), *stdin*);

num\_frames = *atoi*(buf);

*printf*("%d\n", num\_frames);

// *Initialize frame list. page\_number = -1 = free*

num\_free\_frames = num\_frames;

*for* (int i = 0; i < num\_frames; i++)

{

frames[i].page\_number = -1;

}

// *Read in page reference string*

*printf*("Enter page reference string (e.g. 1 2 3 2 1): ");

*fgets*(page\_reference\_string, sizeof(page\_reference\_string), *stdin*);

*printf*("%s\n", page\_reference\_string);

// *Initialize page table*

*for* (int i = 0; i < *PAGE\_TABLE\_SIZE*; i++)

{

page\_table[i].valid = 0;

page\_table[i].frame = 0;

;

}

// *Parse page reference string and simulate paging*

char \*token = *strtok*(page\_reference\_string, " ");

*while* (token != *NULL*)

{

int page\_number = *atoi*(token);

int frame\_number;

page\_references++;

// *If page is not in memory, page fault occurs, try to get a free frame.*

*if* (page\_table[page\_number].valid == 0)

{

page\_faults++;

frame\_number = *get\_free\_frame*(page\_number, page\_references);// *use page\_references as timestamp*

*if* (frame\_number != -1)

{

page\_table[page\_number].valid = 1;

page\_table[page\_number].frame = frame\_number;

*if* (verbose)

*printf*("Page fault at page %d: allocated into frame %d\n", page\_number, frame\_number);

}

*else*

{

*if* (verbose)

*printf*("Page fault at page %d: No Free Frame!\n", page\_number);

}

}

*else*

{

// *Assignment 2*

// *Update timestamp of the referenced page in the frames list*

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].page\_number == page\_number)

{

frames[i].timestamp = page\_references;

}

}

*if* (verbose)

*printf*("Page hit at page %d\n", page\_number);

}

token = *strtok*(*NULL*, " ");

}

// *Calculate page fault rate*

float page\_fault\_rate = (float)page\_faults / page\_references \* 100;

*printf*("Page Fault Rate: %.2f%%\n", page\_fault\_rate);

*return* 0;

}

#*include* <stdio.h>

#*include* <stdlib.h>

#*include* <stdint.h>

#*include* <string.h>

#*define* *PAGE\_TABLE\_SIZE* 128

#*define* *MAX\_FRAMES* 128

typedef struct PageTableEntry

{

uint16\_t valid : 1;

uint16\_t frame : 15;

} PageTableEntry;

typedef struct FrameEntry

{

int page\_number;

int timestamp;

} FrameEntry;

PageTableEntry page\_table[*PAGE\_TABLE\_SIZE*];

FrameEntry frames[*MAX\_FRAMES*];

int num\_frames, num\_free\_frames;

int *get\_free\_frame*(int page\_number, int timestamp)

{

*if* (num\_free\_frames > 0)

{

// *Get the first free frame*

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].page\_number == -1)

{

// *Assignment 1.1*

// *Update page number and timestamp of the free frame*

frames[i].page\_number = page\_number;

frames[i].timestamp = timestamp;

num\_free\_frames--;

*return* i;

}

}

}

// *If no free frame, select one of occupied frames, using FIFO alorithm.*

*else*

{

int oldest\_frame = 0;

int min\_timestamp;

// *Assignment 1.2*

// *Find the oldest frame that is to be replaced*

min\_timestamp = 1 << 30;

*for* (int i = 0; i < num\_frames; i++)

{

*if* (frames[i].timestamp < min\_timestamp)

{

oldest\_frame = i;

min\_timestamp = frames[i].timestamp;

}

}

// *Assignment 1.3*

// *invalidate the replaced page in the page table (valid=0)*

page\_table[frames[oldest\_frame].page\_number].valid = 0;

// *Assignment 1.4*

// *assign page number and timestamp to the selected frame (frames[oldest\_frame])*

frames[oldest\_frame].page\_number = page\_number;

frames[oldest\_frame].timestamp = timestamp;

*return* oldest\_frame;

}

}