//read from stdin

//only n pages loaded at given time

// a page 2^12 = 4kilobytes

//page table 2^20 entries. use array

//page offeset 2^12

//your process has been allocated n frames, n is given on command line

// keep track of number of page faults and print statistics

// use fifo

#include <stdio.h>

#include <stdlib.h>

unsigned int\* createTable();

int getFrame(int, unsigned int [], int []);

int numberOfFaults = 0;

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

int globalTime = 0;

int nFrames = atoi(argv[1]);

unsigned int logicalAddress;

int prevTable, currTable, offset, frame, i, physicalAllocation;

unsigned frameArr[nFrames]; //get no of frames from command line

int presentFrame[nFrames];

unsigned int\* pageTable = createTable(); // 2^20 entries

for(i=0;i<nFrames;i++){

frameArr[i] = -1;

presentFrame[i] = -1;

}

while(!feof(stdin)){

scanf("%ud", &logicalAddress);

currTable = (logicalAddress >> 12) & 0x2FFF;

// printf("%d -> %d\n", logicalAddress, currTable);

// printf("Now the current offset\n");

offset = logicalAddress & 0xFFFF;

// printf("Offset -> %d\n", offset);

if(pageTable[currTable] == 4096){

frame = getFrame(nFrames, frameArr, presentFrame);

if(frameArr[frame] != -1){

prevTable = (frameArr[frame] >> 12)& 0x2FFF;

pageTable[prevTable] = 4096;

numberOfFaults++;

}

pageTable[currTable] = frame;

}else{

frame = pageTable[currTable];

}

frameArr[frame] = logicalAddress;

presentFrame[frame] = globalTime;

globalTime++;

physicalAllocation = (frame << 12) + offset;

printf("This is the Logical Address: %u -> Physical Address: %d\n", logicalAddress, physicalAllocation);

}

printf("------Page Fault statistics-------\n");

printf("Frequency of page faults: %d\n", numberOfFaults);

free(pageTable);

printf("------End of Adress converter------");

return 1;

}

unsigned int\* createTable(){

unsigned int \* ptr = (unsigned int \*) calloc(1048576, sizeof(unsigned int));

//each page has a size of 2^12

int x;

for(x=0; x<1048576;x++){

ptr[x] = 4096;

}

return ptr;

}

int getFrame(int nFramesIn, unsigned int frameArrIn[], int presentIn[]){

int frameToUse = -1;

int frameTobeRemoved = 0;

int z;

for(z=0;z<nFramesIn;z++){

if(frameArrIn[z] == -1){

frameToUse = z;

return frameToUse;

}

}

for(z=0;z<nFramesIn;z++){

int frameToRemove = 0;

if(sizeof(presentIn) == nFramesIn){

presentIn[frameToRemove] = -1;

frameToUse = frameToRemove;

frameTobeRemoved++;

}

}

return frameToUse;

}

Text

Description automatically generated