**Activity 8: Virtual Memory**

ชื่อกลุ่ม เทนโด โซจิ

|  |  |  |
| --- | --- | --- |
|  | ชื่อ - นามสกุล | รหัสนิสิต |
| 1 | วรันธร จันทร์สว่าง | 6432154921 |
| 2 | ศิวภัทร กาญจนะ | 6430376521 |
| 3 | มณธวรรษ สาวะรักษ์ | 6532143021 |

**วัตถุประสงค์**

1. เพื่อให้นิสิตเข้าใจหลักการทำงานของ page fault และ page replacement
2. เพื่อให้นิสิตสามารถเปรียบเทียบการทำงานและคุณสมบัติของ page replacement algorithm แบบต่างๆ

**กิจกรรมในชั้นเรียน**

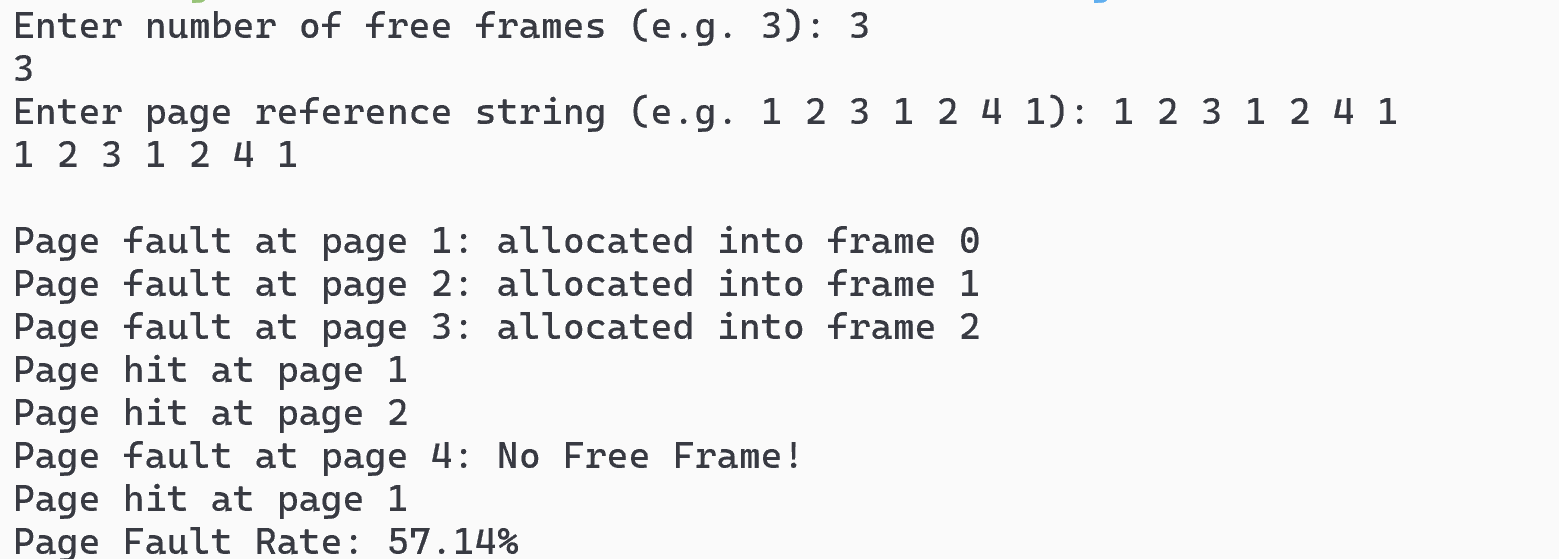
ให้นิสิตศึกษาการทำงานของโปรแกรม pagefault\_noreplece.c ที่ให้ไว้ข้างล่าง

โปรแกรมนี้จำลองการทำงานของ page fault และคำนวณอัตราการเกิด page fault แต่โปรแกรมนี้ยังไม่ได้จัดการกรณีที่ไม่มี frame ว่างเหลือให้ใช้

โปรแกรมนี้เมื่อรันแล้วจะขอให้ผู้ใช้ป้อนข้อมูลสองอย่าง ได้แก่ จำนวน frame ที่มีให้ใช้ และ page reference string

และถ้าตอนรันใส่ option -v จะพิมพ์รายละเอียดของการเกิด page fault ด้วย

ตัวอย่างการใช้งานโปรแกรม pagefault\_noreplace -v เมื่อให้จำนวน frame = 3 และ page reference string = 1 2 3 1 2 4 1



pagefault\_noreplace.c

|  |
| --- |
| // A program that simulates page faults and calculates page fault rate.  // Input: a list of page references (a series of page numbers, separated by a space).  // Output: page fault rate  // Option: -v --> verbose mode: print the result of every page reference,  // whether a page fault occurs, the involved page table entry, page number, and frame number.  #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;  PageTableEntry page\_table[PAGE\_TABLE\_SIZE];  int frames[MAX\_FRAMES];  int num\_frames, num\_free\_frames;  int get\_free\_frame(int page\_number) {  if (num\_free\_frames > 0) {  // Get the first free frame  for (int i = 0; i < num\_frames; i++) {  if (frames[i] == -1) {  frames[i] = page\_number;  num\_free\_frames--;  return i;  }  }  }  else {  return -1; // No free frame available  }  }  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. -1 = free  num\_free\_frames = num\_frames;  for (int i = 0; i < num\_frames; i++) {  frames[i] = -1;  }  // Read in page reference string  printf("Enter page reference string (e.g. 1 2 3 1 2 4 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);  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 {  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;  } |

**สิ่งที่ต้องทำ**

โปรแกรม pagefault\_assignment.c ที่ให้ข้างล่าง เป็นโปรแกรมที่ปรับปรุงมาจากโปรแกรม pagefault\_noreplace เพื่อให้สามารถจัดการกรณีที่ไม่มี frame ว่าง ด้วยการทำ page replacement โดยใช้อัลกอริทึม First In First Out (FIFO) หรือ Least Recently Used (LRU) ซึ่งทั้งสองอัลกอริทึมมีการเก็บข้อมูล timestamp ของแต่ละ frame และเมื่อมีความจำเป็นจะต้องทำ page replacement ก็จะเลือก frame ที่เก่าที่สุด (timestamp น้อยที่สุด) ความแตกต่างของสองอัลกอริทึมนี้อยู่ที่ FIFO จะอัพเดต timestamp เมื่อมีการนำ page ใหม่เข้ามาใน frame ตอนที่เกิด page fault เพียงครั้งเดียว แต่ LRU จะอัพเดต timestamp ทุกครั้งที่มีการเข้าถึงข้อมูล

เพื่อความง่าย โปรแกรมนี้ใช้เพียงอาร์เรย์ชื่อ frames ในการเก็บข้อมูล page\_number และ timestamp การค้นหา frame ที่ต้องการก็สามารถใช้การวนลูป

1. ให้นิสิตเติมโค้ดในส่วนที่มี comment ว่า Assignment 1.x เพื่อทำให้โปรแกรมใช้อัลกอริทึม FIFO
2. ให้นิสิตเติมโค้ดต่อจากโปรแกรมที่ได้ในข้อที่แล้ว ในส่วนที่มี comment ว่า Assignment 2 เพื่อทำให้โปรแกรมใช้อัลกอริทึม LRU

นิสิตสามารถทดสอบความถูกต้องของโปรแกรม เช่น ให้ free frame = 3 และ page reference = 1 2 3 1 2 4 1 หรือตามตัวอย่างในสไลด์ (7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1) และเทียบผลลัพธ์

|  |
| --- |
| // A program to simulates page faults and calculates page fault rate.  // Input: a list of page references (a series of page numbers, separated by a space).  // Output: page fault rate  // Options:  // -v --> verbose mode: print the result of every page reference  // -a <alg> --> choose algorithm: fifo (default) or lru  #include <stdio.h>  #include <stdlib.h>  #include <stdint.h>  #include <string.h>  #include <unistd.h>  #define PAGE\_TABLE\_SIZE 128  #define MAX\_FRAMES 128  typedef struct PageTableEntry {  uint16\_t valid : 1;  uint16\_t frame : 15;  } PageTableEntry;  typedef struct OccupiedFrameEntry {  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 frames[i], and num\_free\_frames  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 the chosen algorithm  else { // all frames are occupied  int oldest\_frame = 0;  int min\_timestamp = frames[0].timestamp;    // Assignment 1.2  // Find the oldest frame that is to be replaced  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)  int victim\_page = frames[oldest\_frame].page\_number;  page\_table[victim\_page].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;  }  return -1; // Should never reach here  }  void print\_usage(const char\* program\_name) {  printf("Usage: %s [-v] [-a alg]\n", program\_name);  printf("Options:\n");  printf(" -v Enable verbose mode\n");  printf(" -a alg Choose algorithm: fifo (default) or lru\n");  }  int main(int argc, char \*argv[]) {  char buf[5];  int page\_faults = 0, page\_references = 0;  char page\_reference\_string[1024];  int verbose = 0;  int use\_lru = 0; // Default to FIFO  int opt;  // Parse command line arguments  while ((opt = getopt(argc, argv, "va:")) != -1) {  switch (opt) {  case 'v':  verbose = 1;  break;  case 'a':  if (strcmp(optarg, "lru") == 0) {  use\_lru = 1;  } else if (strcmp(optarg, "fifo") == 0) {  use\_lru = 0;  } else {  fprintf(stderr, "Invalid algorithm: %s\n", optarg);  print\_usage(argv[0]);  return 1;  }  break;  default:  print\_usage(argv[0]);  return 1;  }  }  // Read in number of free frames  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 1 2 4 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;  }  printf("Using %s algorithm\n", use\_lru ? "LRU" : "FIFO");  // 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 {  // For LRU, update timestamp on page hits  if (use\_lru) {  // Assignment 2  // Update timestamp of the referenced page in the frames list  int frame\_num = page\_table[page\_number].frame;  frames[frame\_num].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;  } |

**สิ่งที่ต้องส่งใน MyCourseVille**

1. ไฟล์โปรแกรมที่แก้ไขแล้วสำหรับ pagefault\_assignment.c
2. capture หน้าจอผลลัพธ์ เมื่อรันโปรแกรมแบบ verbose และใช้ page replacement algorithm แบบ fifo และ lru
3. อธิบายเปรียบเทียบผลลัพธ์ของ fifo และ lru

จะใส่สิ่งที่ต้องส่งโดยเพิ่มลงในไฟล์นี้ หรือส่งเป็นไฟล์แยกต่างหากก็ได้

สิ่งที่ส่ง

1. แนบไปพร้อม เอกสารฉบับนี้

2.

A screenshot of a computer program

AI-generated content may be incorrect.

3. ผลลัพธ์:

FIFO: Page Fault Rate สูงกว่าเนื่องด้วย เมื่อต้องการใช้ 1 อีกครั้งหนึ่ง ถูก 4 แทนที่ไป (เพราะ 1 เข้ามาเป็นอันดับแรก)

LRU: Page Fault Rate ต่ำกว่าเนื่องด้วย ในกรณีเดียวกันกับข้างบน 1 ยังคงอยู่ใน Frame ทำให้คำสั่งสุดท้ายยังคงเกิด Page Hit (Frame ที่ถูกแทนที่ไปโดย 4 คือ Frame ที่ 3 อยู่ เนื่องด้วย 3 ไม่เคยถูกเรียกใช้เลยหลังจากเข้ามาครั้งแรกต่างจาก 1 และ 2 ที่ยังถูกเรียกใช้ จึงทำให้ 3 ถูกแทนที่ไป)