מכללה אקדמית הדסה
החוג למדעי המחשב
מועד הגשה: 15.05.2011

מערכות הפעלה ב'
תרגיל "7" בנושא : Virtual memory

רקע:
בתרגיל זה נסמלץ מערכת פשוטה (מאוד) לניהול זיכרון במחשב של תהליך בודד. אשר משתמשת במנגנון demand-paging תוך מתן דגש על יכולת להשתמש בדיסקים כהרחבה של הזיכרון.

אנו נכתוב מעין מערכת לניהול זיכרון ונממש את הפונקציות load, store

הגדרת התרגיל:
הערה: הprototype המדיוק של מבנה הנתונים והפונקציות שיש לממש נמצא בהמשך.

לצרכי הפשטה נאפשר במערכת שלנו הרצה של תוכנית (תהליך) בודד בלבד. מערכת מחשב שאותה נסמלץ תהיה מערכת 10bit, גודל זיכרון ראשי של 64 כאשר גודל כל דף הוא $2^4 = 16$.
וישנם $2^6 = 64$ דפים ו $2^2 = 4$ מסגרות בזיכרון. גודל דיסק– ע"פ שיקול דעתכם.
כול תהליך יורכב מ 4 חלקים: text, data, bss , heap. נניח (שוב, לצרכי הפשטה) שכל כתובת וירטאולית מ 0 עד text+data+bss+heap היא חוקית, כל עוד text+data+bss+heap קטן מ 2^{10}.

תזכורת: BSS **אינו** שמו של מקטע המחסנית, אלא של המקטע המכיל את המשתנים הסטאטיים והגלובליים הלא מאותחלים (ועל כן אינם נשמרים בקובץ הבינארי).

מנגנון לניקוי הדפים
יש לנקות את הדפים מהזיכרון ע"י אלגוריתם FCFS.

Struct SIM_DATABASE
מבנה הנתונים העיקרי בתוכנית הוא הstruct בשם SIM_DATABASE הכולל את השדות הבאים:

- **Page_table** - טבלת הדפים. (מערך של struct – ראה בהמשך)
- **swap_file, swap_fd** - שם הקובץ ה SWAP (יקרא תמיד "swap_file") וה-file descriptor שלו בהתאמה.
- **program_file, program_fd** – שם הקובץ של התוכנית עצמה והfile-descriptor שלה בהתאמה. (זה הוא הexecutable) -> ערך זה יתקבל מהMain של התוכנית ושמו יהיה ע"פ שיקול דעתכם.

struct page_descriptor

struct זה מייצג כניסה בטבלת הדפים והוא כולל את השדות הבאים :

- Valid : 1 – הדף שוכן בזיכרון הראשי. 0 – אחרת.
- Permission : 1 - לדף זה ישנן הרשאות כתיבה וקריאה. 0 – רק הרשאות קריאה
- Touched : 0 – לא נכתבו נתונים לדף זה. 1 - נכתבו נתונים בדף זה.
- Frame –נשתמש בשדה זה ע״מ לדעת היכן הדף בזיכרון\swap.

יש לממש את הפונקציות :

vm_constructor

```
/*
 * Initialize vm, set function pointers to the specific algorithms
 * arguments:
 *      executable - the program file name
 *      text       - the size of the text segment ( pages )
 *      data       - the size of the data segment ( pages )
* return value: A pointer to the sim_database
 */
sim_database_t* vm_constructor( char* executable ,  unsigned short text, unsigned short data,
unsigned short bss) ;
```

- **Executable** - מסוג *char פרמטר זה יכיל שם של תוכנית שאותה נרצה ל״הריץ״ במחשב הוירטואלי שלנו (כזכור, מחשב זה מסמלץ רק את מערכת ניהול הזיכרון). שימו לב : קובץ זה צריך להיות קיים במערכת הקבצים (קובץ זה איננו וירטואלי). יש לכלול בקובץ תווים כך שתוכנו יהיה משמעותי, ואפשר יהיה לעקוב אחרי מה נקרא, ומה הם השינויים המתחוללים בנתונים שנקראו.
- שאלה ? : איך נקרא ונכתוב לקובץ ? ראה בקובץ ששלחתי לכם FileOperation.pdf.

ארגומנטים נוספים הם, הגדלים של :

- **text** – גודל הקוד
- **Data** – משתנים
- **bss** –
-

פונקציה זו תבצע :

- תקצה ותאתחל את ה ptable
- תפתח swap_file בגודל המתאים.
- תפתח את ה program_file -- הוא ה executable.
  - ועוד מבנה נתונים ומשתנים נוספים שתחליטו להקצות – בהתאם למימוש שלכם.

פונקציה זו תחזיר מצביע למבנה הנתונים SIM_DATABASE שאותחל.

vm_load

.

```
/*
 * Load the value in addr 'virtual_addr' into 'value'.
 */
Char value = vm_load( sim_database_t* sim_database, unsigned short virtual_addr) ;
```

- **Sim_database** – מצביע למבנה הנתונים המרכזי.
- **virtual_addr** – כתובת ממנה יש לקרוא. כתובת חוקית היא בטווח של 0 עד (text+data+bss+heap).
- **Value** – את הערך שנמצא בכתובת הרלוונטית נחזיר

שימו לב: נכתוב ונטען char אחד בכול פעם.
כאשר, את מספר הדף בו נמצאת הכתובת המבוקשת נחלץ מיתוך 6 הביטים הראשונים ובעזרת טבלת הדפים (תזכורת: 6 הביטים הראשונים הם מספר הדף, וארבעת האחרונים הם הסטה בדף). את מיקום הדף (בזיכרון\ בSWAP \ לא נטען כלל) נוכל למצוא זאת בטבלת הדפים. אם הדף לא בזיכרון יש לטעון אותו לשם (מה swap או מהexecutable).

vm_store.

```
/*
 * Store 'value' into 'virtual_addr'.
 */
status_t vm_store( sim_database_t* sim_database, unsigned short virtual_addr, unsigned char value ) ;
```

- **Sim_database** – מצביע למבנה הנתונים המרכזי.
- **virtual_addr** – כתובת בה נרצה לכתוב. כתובת חוקית היא בטווח של 0 עד (text+data+bss+heap).
- **Value** – את הערך value נשמור בכתובת הרלוונטית.

ראה הערת מימוש בפונקצית vm_load.

vm_destructor

```
/*
 * Free everything.
 */
void vm_destructor( sim_database_t *sim_database ) ;
```

תשחרר את כל הזיכרון שהיה בשימוש בתוכנית – זו פונ' מאוד פשוטה אין להסתבך

vm_print.

```
/*
 * print everything
 */
status_t vm_print ( sim_database_t* sim_database  ) ;
```

- פונ' זו תדפיס את מבנה הנתונים, תכולת הזיכרון, והסטטיסטיקות השונות, בצורה החביבה אליכם, אך שהמשתמש יבין....

מבנה נתונים שיש להגדיר:

```
*/

---

    */
     * Struct page_descriptor : An entry in the page table
    /*

typedef struct page_descriptor {
        unsigned int     valid        ;                /* is the page in physical memory/*
        unsigned int    touched    ;            /* was the page changed or not/*
        unsigned int    permission;           /* load_only /  load_store/*
         unsigned int     frame;        /* the physical frame number/*

} page_descriptor ;

---

struct sim_database    {
            page_descriptor ptable[64] ;      // pointer to the page table

           sim_sizes sizes ;                  // simulation sizes

           char *program_file;            // The name of the executable
           int program_fd;                // fd to the executable/*

           char *swap_file;               // The name of the swap file
           int swap_fd;                   // fd to the swap file


}
```

The program Main (the "CPU")

```
#define TEXT_SIZE 256         //size of text segment of executable
#define DATA_SIZE 256         //size of data segment of executable
#define BSS_SIZE 256          //size of data segment of executable
#define LETTER_START 65       //first capital letter in ascii table
#define DIFF 25                     //range of capital letters in
ascii table
#define EXEC_FILE "exec"
#define LOOPS 200
char* val;

int main() {
      srand(time(NULL));
            sim_database* db = vm_constructor(EXEC_FILE, TEXT_SIZE,
            DATA_SIZE, BSS_SIZE);
      int i; unsigned short addr;         //virtual address

      for (i = 0; i < LOOPS; i++) {
        addr = rand() % EXEC_SIZE;
        *val = vm_load(db, addr);
        *val = rand() % DIFF + LETTER_START;    //capital let.s
        vm_store(db, addr+1, val);
      }
      vm_print(db);
      vm_destructor(db);
}
```

(3%)

יש להוסיף main נוסף כאשר מגישים את התרגיל שבו לא יהיו מספרים אקראים

בהצלחה