מערכות הפעלה תרגול 8

Shared memory

מתרגל-יורם סגל
yoramse@colman.ac.il

שני אלקובי
פריאל לוי

Process activities

Generate process

Process

- Fork(),
- Exec(),
- Wait(),
- Exit()

Messages between process

Signals

- Signal(),
- Kill(),
- Pause(),
- Alarm()

Data between process

Process Communication

- Files
- Pipe()
- Fifo()

Shared memory between process

Shared Memory

- shm**get**() יצור
- shm**at**() חיבור
- shm**dt**() ניתוק
- shm**ctl**() בקרה
- ftok() מפתח יחודי

T8 - Contents

- PIPE vs. FIFO
- Virtual and physical memory
- Page and Frame
- Shared memory (system calls):
  shm**get**() - get shared memory
  shm**at**() - Attaches the shared memory segment
  shm**dt**() - Detaches the shared memory segment
  shm**ctl**() - Control shared memory

Pipe vs. FIFO

- pipes (צינורות) הם ערוצי תקשורת חד-כיווניים ואילו FIFO הוא ערוץ תקשורת דו-כיווני, כלומר ניתן לבצע הן קריאה והן כתיבה ל-FIFO דרך אותו descriptor

- Pipes אינם מופיעים בהיררכיה של מערכת הקבצים לעומת FIFO שמופיע במערכת הקבצים בשמו.

- לאחר סיום השימוש ב-pipe מצד כל התהליכים (סגירת כל ה-descriptors) מפונים משאבי ה-pipe באופן אוטומטי. ואילו FIFO אינו משוחרר אוטומטית לאחר שהמשתמש האחרון בו סוגר את הקובץ, ולכן יש לפנותו בצורה מפורשת.

זכרון פיזי

- הזיכרון הפיזי הוא זיכרון הנמצא בהתקנים הפיזיים של המחשב. הזיכרון הפיזי מורכב מ:

  - זיכרון פיזי ראשי – RAM - מאכסן את התכנית לביצוע והנתונים שהתכנית משתמשת בהם (נדיף).

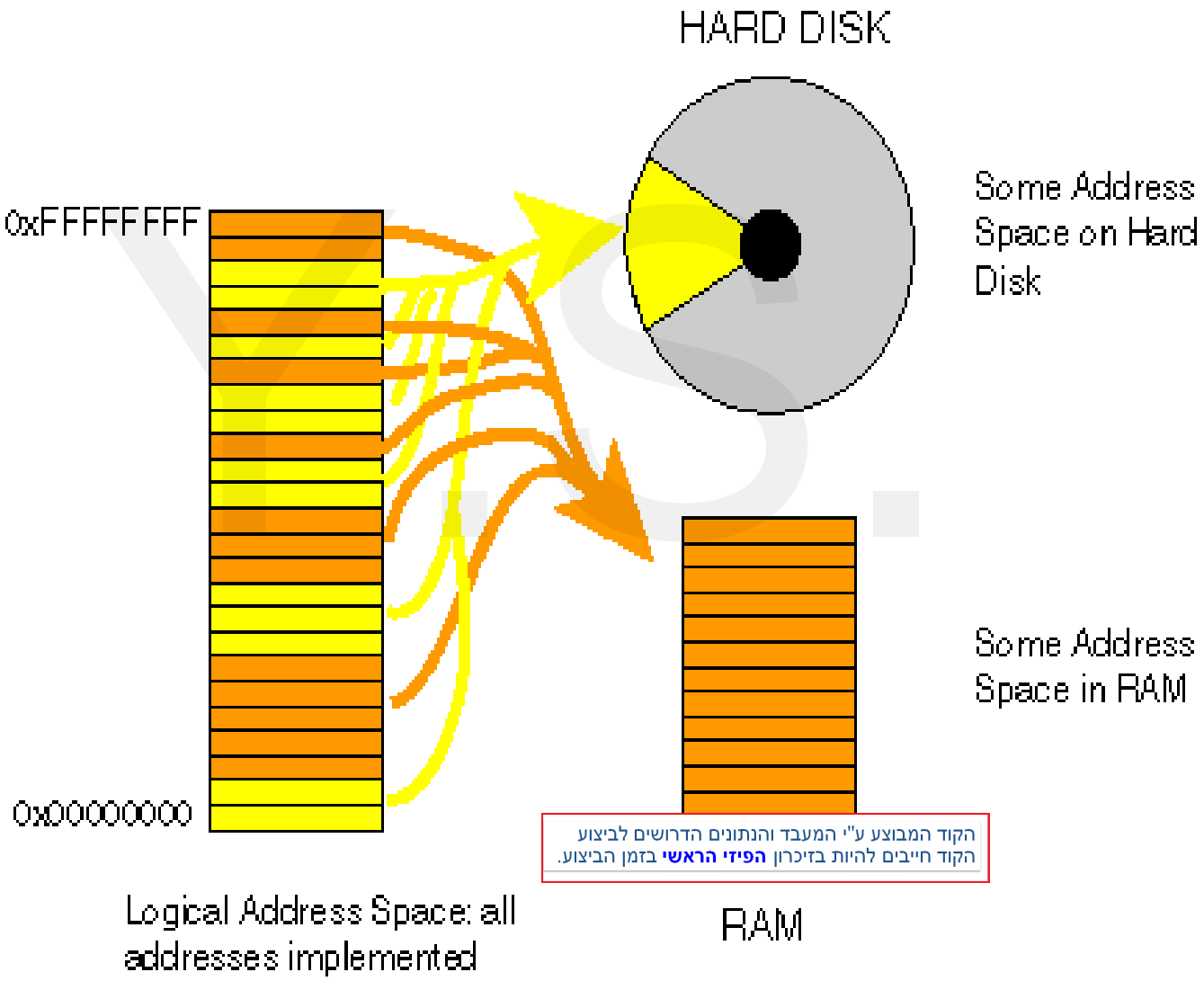
  - זיכרון פיזי משני – דיסק קשיח – (hard disk), מאכסן בדרך כלל כמות גדולה של נתונים, מורכב מדסקיות (לא נדיף).

זכרון וירטואלי ופיזי

- זיכרון וירטואלי הוא מרחב זיכרון מדומה העומד לרשות תהליך.
  - מרחב הזיכרון הוירטואלי של תהליך יכול להיות גדול מהזיכרון הפיזי **הראשי**.
- מרחב הזיכרון הוירטואלי ממופה בחלקו לזיכרון הפיזי הראשי (RAM) ובחלקו לזיכרון הפיזי המשני (HD).

הקוד המבוצע ע"י המעבד והנתונים הדרושים לביצוע הקוד חייבים להיות בזיכרון **הפיזי הראשי** בזמן הביצוע.

זכרון וירטואלי ופיזי



למה צריך זכרון וירטואלי?

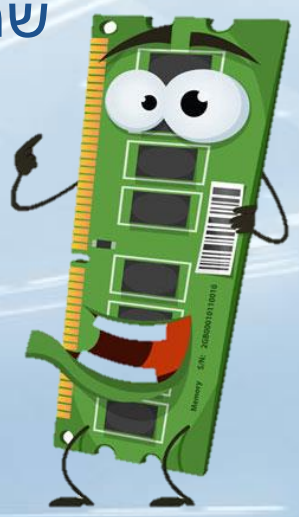
1. הפרדה בין תהליכים:

- לכל תהליך מוגדר מרחב זיכרון וירטואלי משלו, בלתי תלוי בתהליכים אחרים.
- כלומר, מערכת ההפעלה מגינה עלינו בכך שהיא מפרידה את מרחב הכתובות של כל תהליך מכל האחרים ובזכות זאת תהליך אחד לא יכול לפגוע בנתונים של תהליך אחר.

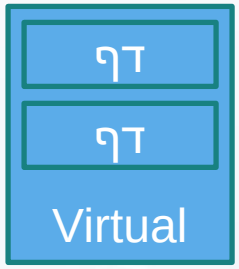
למה צריך זכרון וירטואלי?

2. הגדלת הזיכרון העומד לרשות המערכת:

- בכל נקודת זמן רק חלק ממרחב הזיכרון של תהליך נמצא בזיכרון הפיזי הראשי.
- מאפשר למחשב להריץ מספר רב של תהליכים שסך כל הזיכרון שלהם גדול מהזיכרון הפיזי הראשי (או תהליך בודד שמרחב הזיכרון שלו גדול מהזיכרון הפיזי הראשי).



מרחב זיכרון וירטואלי

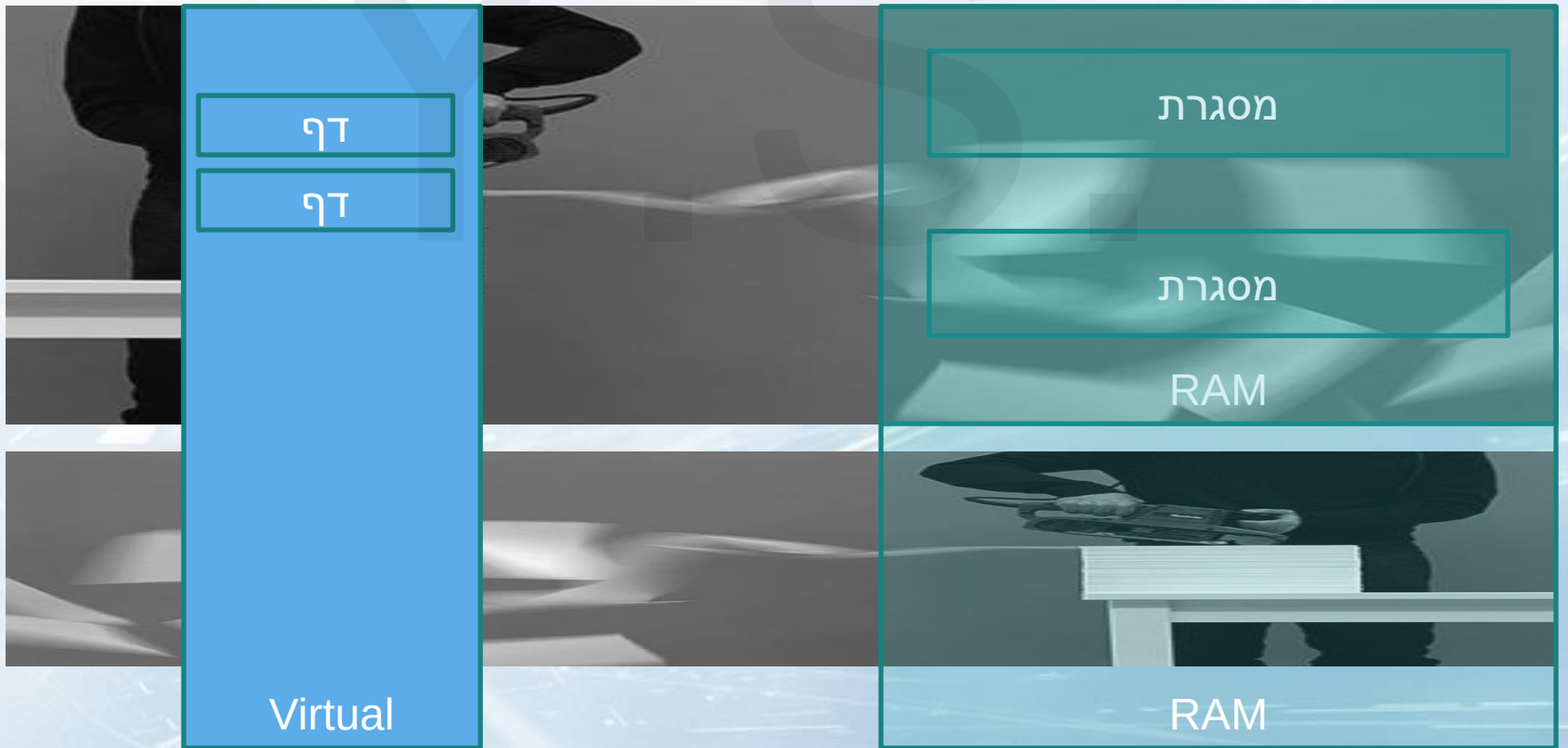


- לכל תהליך מרחב זיכרון וירטואלי משלו.
  - מתחיל בכתובת 0 ומסתיים בכתובת 3GB-1. (המוסכמה בx86)
- מרחב הזיכרון הווירטואלי של תהליך מחולק בכפולות של **דפים**
  - קטעי זיכרון עוקבים בעלי גודל קבוע (בד"כ 4KB)
- תהליך מגדיל את מרחב הזיכרון הוירטואלי שלו ע"י בקשות להקצאת זיכרון.

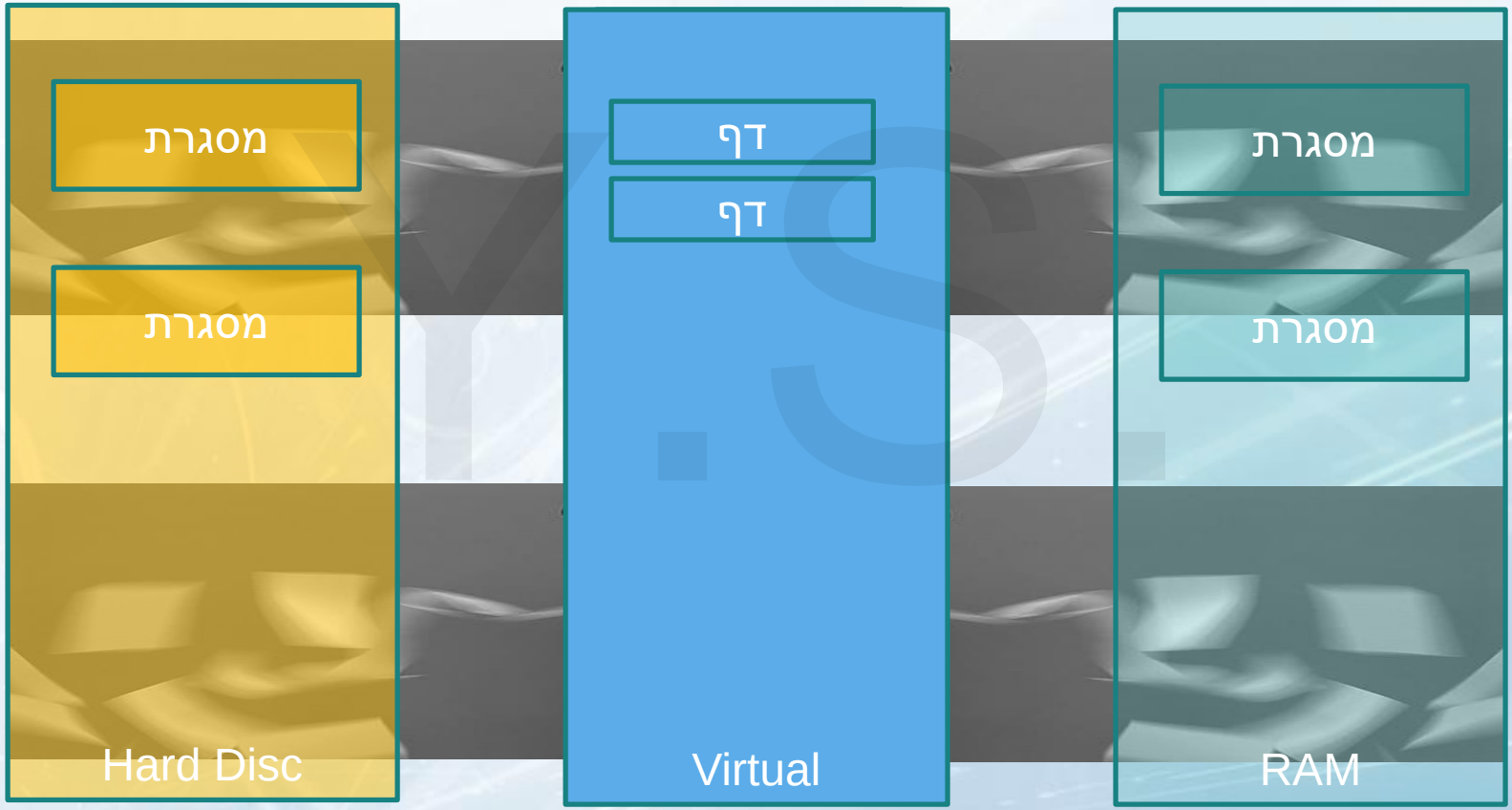
- הזיכרון הפיזי הראשי של המחשב מחולק **למסגרות**, כל-אחת בגודל דף.

מרחב זיכרון וירטואלי

- לביצוע מקטע קוד,
- הדף הרלוונטי מהזיכרון הווירטואלי
- ישובץ לתוך מסגרת של זיכרון הראשי(RAM)



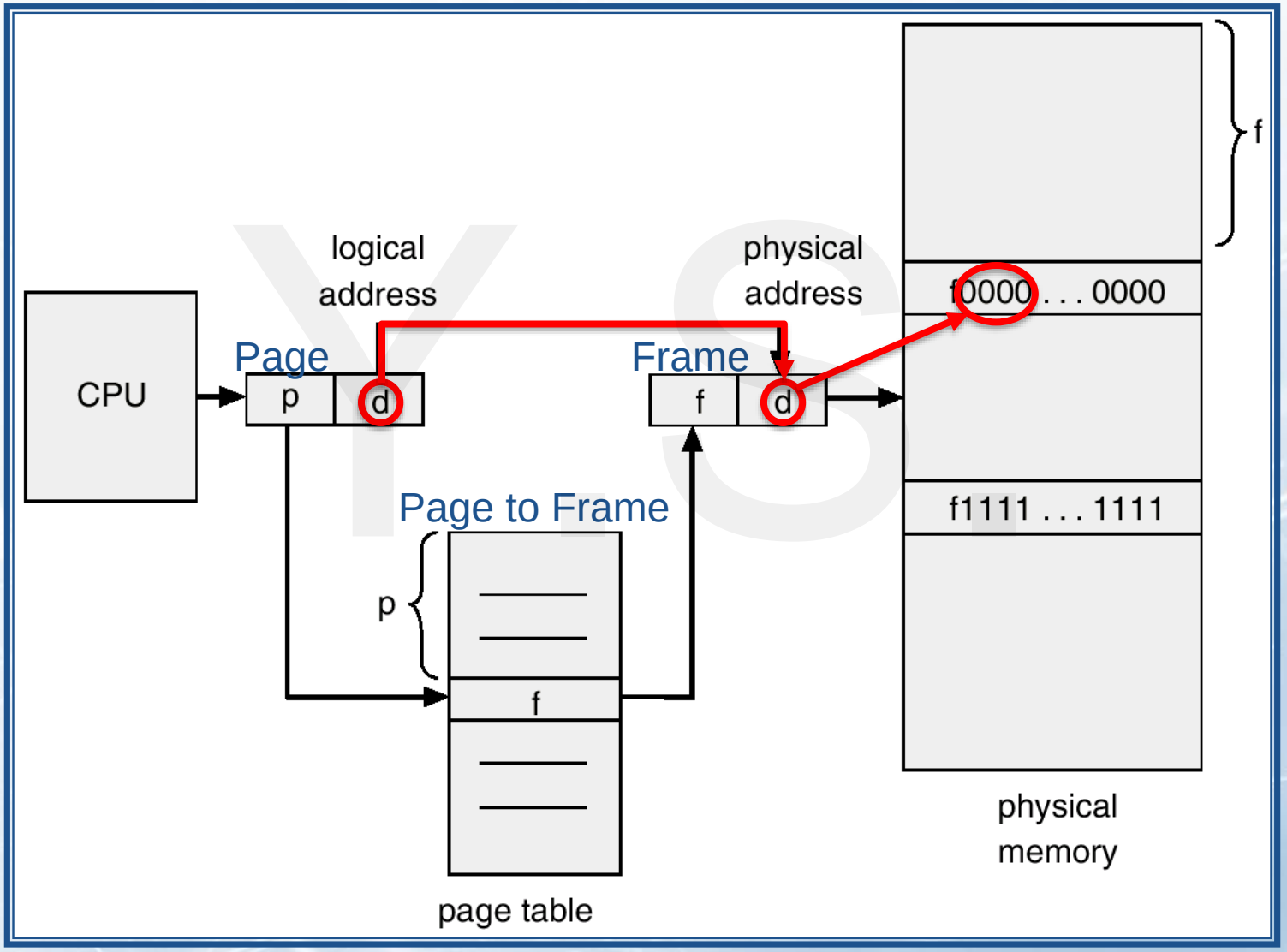
מרחב זיכרון וירטואלי



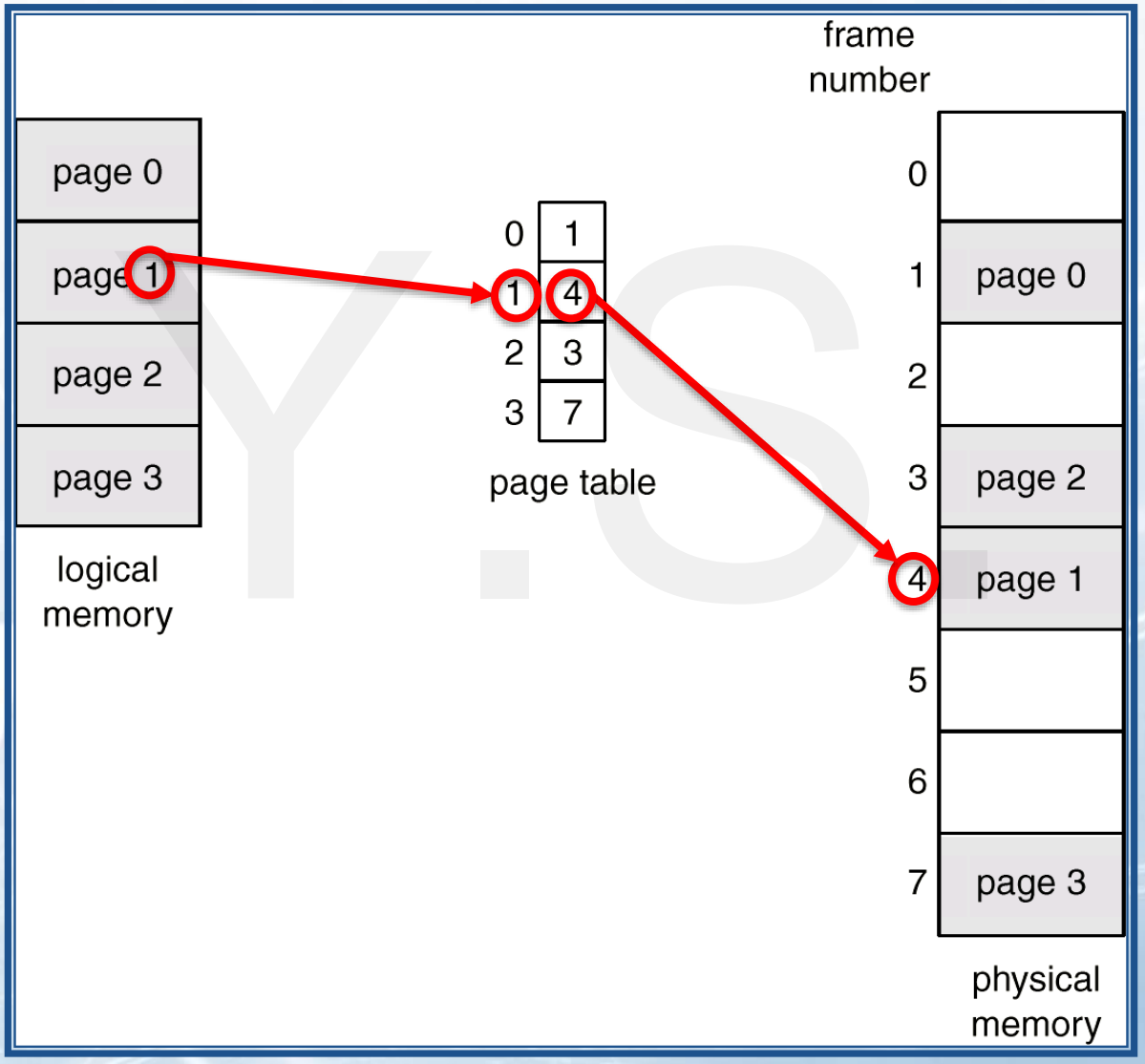
טבלאות דפים

- לכל תהליך יש טבלת דפים משלו (שמנוהלת על ידי מערכת ההפעלה) אשר מנהלת את הדפים שלו.
  - האם הדף נמצא בזיכרון הראשי ובאיזו מסגרת.
- הטבלה מכילה כניסה עבור כל דף במרחב הזיכרון של תהליך.
- כל כניסה של דף בטבלת הדפים מכילה דגל מיוחד (present) המציין האם הדף נמצא בזיכרון הראשי.
  - אם כן, הכניסה מכילה את מספר המסגרת בה מאוחסן הדף בזיכרון הראשי.
  - אם לא, הכניסה מצביעה על המיקום של הדף בדיסק.

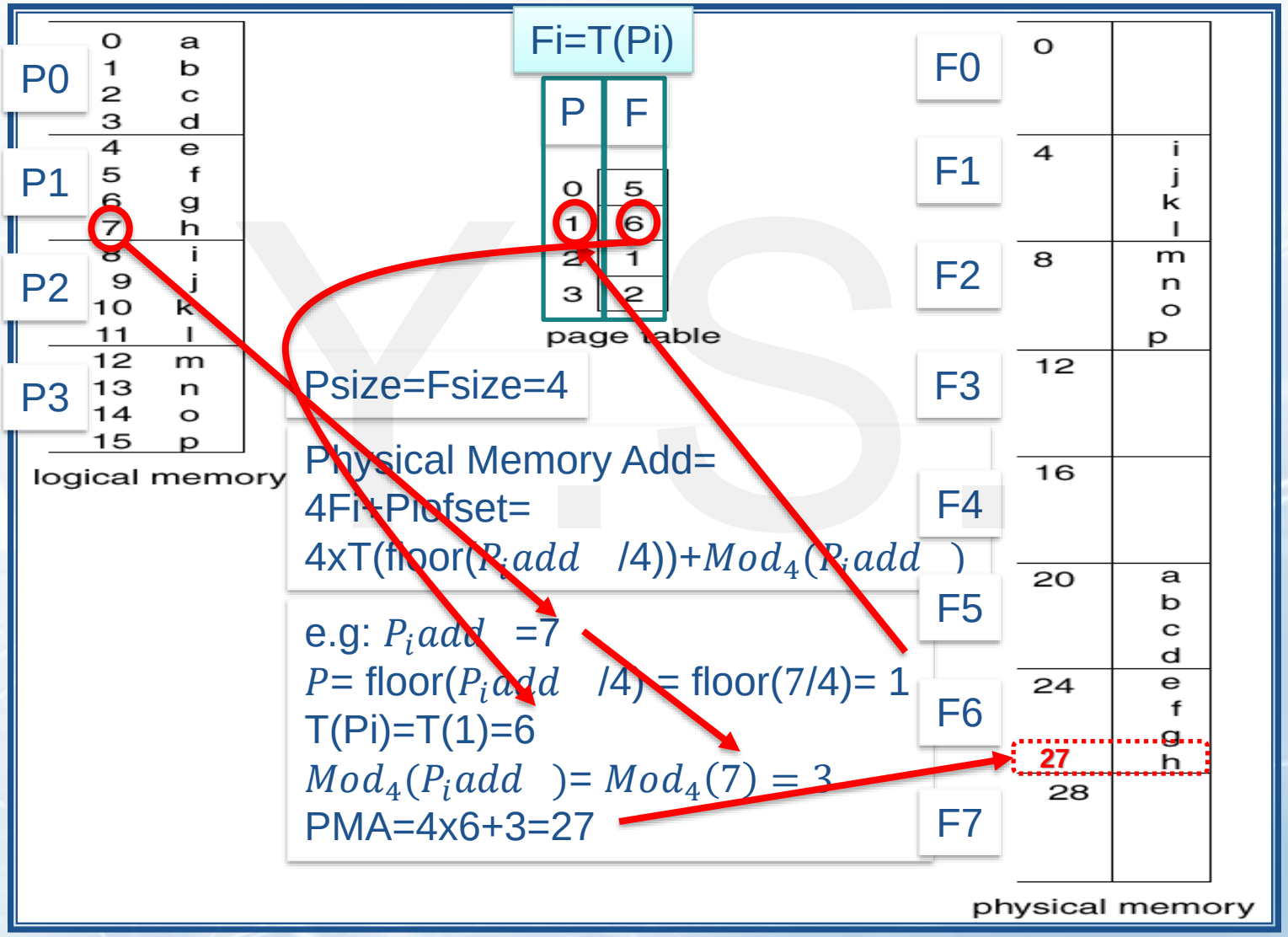
תרגום כתובת וירטואלית



תרגום כתובת וירטואלית - דוגמא



תרגום כתובת וירטואלית – דוגמא מפורטת



זכרון משותף(IPC) Inter Process Communication

- מה הבעיה העיקרית ב pipes או בFIFO?
  - זמן... התערבות של מערכת ההפעלה בכל קריאה/כתיבה.
- ניתן להגדיר אזור זיכרון שיהיה משותף למספר תהליכים.
- המנגנון המאפשר זאת נקרא:

IPC Shared Memory

- למעשה נמפה את אותה מסגרת למספר תהליכים

יתרונות זכרון משותף

- מאפשר למספר תהליכים לגשת לאותו חלק בזכרון ולחלוק את אותו המידע, בניגוד ל pipe שקריאה ממנו מרוקנת את המידע.
  - כל התהליכים רואים את המידע שהשתנה.
  - מהיר יותר מ pipes.
    - לא מצריך system call בזמן קריאה/כתיבה.

יצירת זיכרון משותף ושימוש בו

1. יצירת מפתח ייחודי.
2. שימוש במפתח על מנת להקצות חלק ספציפי בזיכרון.
3. התחברות של כל התהליכים הרלוונטיים לאזור המשותף בזיכרון.

shmget() (get shared memory)

- #include <sys/ipc.h>
  #include <sys/shm.h>
- **int shmget(key_t *key*, size_t *size*, int *shmflg*);**

  key- מפתח זיהוי או הערך IPC_PRIVATE(בשביל מקום כלשהו חדש)

  size- גודל רצוי (יעוגל לגודל דף)

  shmflg – הרשאות | *IPC_CREAT(ליצור מקום חדש עבור המפתח) | IPC_EXCEL (כשלון אם הסיגמנט קיים)*

  ערך מוחזר: *shmid-* ערך מספרי המזהה באופן ייחודי שטח שנוצר לשיתוף או 1- עבור כשלון

יעוד הפונקציה: shmget()

- #include <sys/ipc.h>
  #include <sys/shm.h>
- **int shmget(key_t *key*, size_t *size*, int *shmflg*);**

קריאת מערכת שנועדה:

1. להקצות אזור משותף בזיכרון (על ידי שימוש בkey שאף תהליך אחר עוד לא נתן או על ידי שימוש בIPC_PRIVATE)

או

2. לקבל מזהה של אזור בזיכרון שתהליך אחר כבר הקצה(על ידי שימוש בkey שתהליך אחר כבר השתמש בו על מנת להקצות זיכרון משותף).

יצירת זכרון משותף t8_1.c

```
#define SHM_SIZE 4096
#define FLAGS IPC_CREAT | 0644

int main()
{
    int shm_id;
    shm_id=shmget(IPC_PRIVATE,SHM_SIZE,FLAGS);
    if(shm_id==-1)
    {
        printf("error creating shared memory\n");
        exit(0);
    }
    printf("the shared memory segment ID is: %d\n",shm_id);
}
```

```
raz@ubuntu:~/rmi/os2018/T7$ ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 294912     raz        600        524288     2          dest
0x00000000 524289     raz        600        16777216   2
0x00000000 425986     raz        600        524288     2          dest
0x00000000 1179651    raz        600        16777216   2          dest
0x00000000 786436     raz        600        524288     2          dest
0x00000000 884741     raz        600        524288     2          dest
0x00000000 983046     raz        600        524288     2          dest
0x00000000 1146887    raz        600        524288     2          dest
0x00000000 5013512    raz        600        268435456  2          dest
0x00000000 5767177    raz        600        2097152    2          dest
0x00000000 2359306    raz        600        524288     2          dest
0x00000000 1572875    raz        600        524288     2          dest
0x00000000 1671180    raz        600        524288     2          dest
0x00000000 5799949    raz        666        4096       0
0x00000000 5111822    raz        600        29884416   2          dest
0x00000000 5046288    raz        600        31395840   2          dest

raz@ubuntu:~/rmi/os2018/T7$
```

סוגי שגיאות

1.ביקשנו גודל שקטן מהמינימום (SHMMIN) או גדול מהמקסימום (SHAMMAX) המותרים.

2. האובייקט כבר קיים והעברנו

IPC_CREAT|IPC_EXCL

3. האובייקט לא קיים (ולא הדלקנו את IPC_CREATE)

4.המערכת לא הצליחה להקצות זיכרון.

shmat()

Attaches the shared memory segment

- #include <sys/types.h>
- #include <sys/shm.h>
- void *shmat(int shmid, const void *shmaddr, int shmflg);

מטרה: לקשר את הזכרון המשותף לתהליך.

- shm_id: המזהה של הזכרון המשותף (הערך שחזר מהפונקציה shmget()).
- shmaddr: לרוב NULL, או כתובת הסט מפורשת למיפוי השטח הזיכרון המשותף
- *shmflg: מספר פרמטרים מופרדים ע"י |*

SHM_RDONLY: יפתח את השטח לקריאה בלבד (ברירת המחדל היא R+W)

SHM_RND: אם הוכנס שטח זיכרון, flag זה יעגל אותו למטה לכתובת חוקית שהינה כפולה של גודל דף.

A successful shmat() call updates the members of the shmid_ds structure (see shmctl()) associated with the shared memory segment as follows:

- shm_atime is set to the current time.
- shm_lpid is set to the process-ID of the calling process.
- shm_nattch is incremented by one.

shmat()

עוד על ... shmat

- הפעולה מגדילה מונה פנימי (בדסקריפטור) שסופר את מספר ה-attachments (שימושי לפעולת ה-delete)
- ערך מוחזר: המצביע לזכרון – במקרה הצלחה, או 1- במקרה של כשלון
- שגיאות אפשריות:
  - אין הרשאות
  - קונפליקט בכתובות
  - בעיה בהקצאת הזכרון

shmdt()

Detaches the shared memory segment

- #include <sys/types.h>
- #include <sys/shm.h>
- int shmdt(const void *shmaddr);

- *shmaddr* – כתובת מפורשת אליה ממופה שטח הזיכרון המשותף

- הפקודה מנתקת את הקשר בין התהליך לבין הזיכרון המשותף (לא מוחקת את הזיכרון)
- כאשר התהליך מסתיים הניתוק מתבצע אוטומטית
  - (כי אין תהליך....)

קריאה/כתיבה לזכרון משותף t8_2.c

```
#define SHM_ID 720911 // same SM's id from t8_1

int main()
{
    char* shmaddr;
    if((shmaddr=shmat(SHM_ID,0,0))==(char*)-1)
    {
        printf("error in attaching to the shared memory\n");
        exit(0);
    }
    printf("the shared memory contains: %s\n",shmaddr);
    strcpy(shmaddr,"hello world");

    if(shmdt(shmaddr)==-1)
        printf("detaching error\n");
}
```

יצירת זכרון משותף – מפתח יחודי t8_3.c

איך אפשר לייצר מפתח מסוג: key_t ?

בעזרת הפקודה ftok

key_t ftok(char ****pathname*, char** *proj_id***);**

pathname- שם של קובץ

The specified pathname must specify an existing file that is accessible to the calling process or the call will fail

proj_id- אות שאתם בוחרים

The second argument (proj_id) exists simply so you can create a bunch of IPC resources

Combines 16 bits from inod number, 8 bits from device and 8 bits of proj_id

t8_3.c

- הקוד מחייב כי הקובץ shmdemo.c יהיה נגיש לקוד הרץ
- בשורת הפקודה יש להוסיף טקסט אשר יוטמע בתוך הזכרון המשותף typethis 3.out/.

זכרון משותף – אבא ובן t8_4.c

- fork()
  - נוצר מרחב כתובות וירטואלי אצל הבן שבו אותה כתובת וירטואלית (בעותק נפרד) מצביעה על אותה מסגרת.
- exec() וגם exit()
  - שטחי הזיכרון עוברים detache.
- מדוע השטחים לא מושמדים?
  - יתכן ותהליך אחר יעלה ויעשה שימוש בשטחים האלו.
- t8_4.c

Wait() - Wait till child process will change its state: child terminated, child was stopped by a signal, child was resumed by a signal

shmctl() - control: מחיקה/עדכון

```
#include <sys/shm.h>
int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

- פעולה:
  - מאפשר שליפה ועדכון של פרמטרים של השטח המשותף, מאפשר השמדה
- פרמטרים:
  - *shmid* – מזהה האובייקט (קבלנו אותו מ shmget)
  - *cmd* – (אופרטור יחיד!)
    - **IPC_STAT** – מאפשר שליפה לתוך *buf* של הסטאטוס של השטח
    - **IPC_SET** – מאפשר קביעה של פרמטרי הסטאטוס של השטח ע"פ אלו שמוגדרים ב *buf* ישנה את הפרמטרים הבאים: shm_perm.uid shm_perm.gid shm_perm.mode
    - **IPC_RMID** – מורה על השמדה (לאחר הניתוק האחרון, הרשאות משתמש נבדקות – רק לowner, לcreator או למי שיש לו את ההרשאות המתאימות)
  - *buf* – מבנה הנתונים אליו יקרא / ממנו יילקחו הסטאטוס של השטח (פרטים בשקף הבא)
- ערך מוחזר
  - 0 בהצלחה
  - 1- בכישלון

מחיקה/עדכון - shmctl()

שיחרור הזכרון המשותף

- כדי לשחרר את שטח הזיכרון יכול כל תהליך שיש לו הרשאת קריאה + כתיבה על השטח לבצע:

```
if (shmctl(shm_id, IPC_RMID, NULL) == -1)
{
    perror("shmctl failed") ;
    exit(EXIT_FAILURE) ;
}
```

- קטע זכרון משותף שלא שוחרר מסיבה כלשהי (למשל: התהליך עף...) ממשיך להתקיים (ולגזול משאבי מערכת).
  - בפרט, ניסיון הקצאה שני (הרצה חוזרת של תוכניתנו) – ההקצאה תכשל

shmid_ds

```
struct shmid_ds {
    struct ipc_perm shm_perm;/* operation perms */
    int shm_segsz;          /* size of segment (bytes) */
    time_t shm_atime;       /* last attach time */
    time_t shm_dtime;       /* last detach time */
    time_t shm_ctime;       /* last change time */
    unsigned short shm_cpid;        /* pid of creator */
    unsigned short shm_lpid;        /* pid of last operator */
    short shm_nattch;       /* no. of current attaches */
    ...
};

struct ipc_perm {
    key_t  key;
    ushort uid;     /* owner euid and egid */
    ushort gid;
     .
     .
     .
    ushort mode;
    …
};
```

עבודה מה shell

מציג את משאבי הזיכרונות המשותפים : ipcs –m

```
root@fedora:~
[root@fedora ~]# ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
0x00000000 262144     root       600        393216     2
0x00000000 32769      root       644        40         2
0x00000000 65538      root       644        16384      2
0x00000000 98307      root       644        268        2
0x00000000 294916     root       600        393216     2
0x00000000 327685     root       600        393216     2
0x00000000 360454     root       600        393216     2
0x00000000 393223     root       600        393216     2
0x00000000 425992     root       600        393216     2
0x00000000 458761     root       600        393216     2
0x00000000 491530     root       600        393216     2
0x00000000 524299     root       600        393216     2
```

To delete shared memory from the shell use:

- ipcrm shm shmid
- ipcrm shm 14092

דוגמה shmctl

- t8_5.c הדפסת מידע של הזכרון המשותף
- t8_6.c מחיקת זכרון משותף
- t8_7.c t8_8.c דוגמא כוללת של קריאה וכתיבה.