Operating Systems – 234123

**Homework Exercise 4 – Dry**

|  |  |  |
| --- | --- | --- |
| **Omer Taub** | [**Omer.taub@campus.technion.ac.il**](mailto:Omer.taub@campus.technion.ac.il) | **205761661** |
| **Shahar Or** | [**Shahar.or@campus.technion.ac.il**](mailto:Shahar.or@campus.technion.ac.il) | **205735848** |

Teaching Assistant in charge:

**Sami Zreik**

Assignment Subjects & Relevant Course material

**Virtual Memory & IPC**

**Recitations 7 & 10, Lecture 6**

# Submission Format

1. Only **typed** submissions in **PDF** format will be accepted. Scanned handwritten submissions will not be graded.
2. The dry part submission must contain a single PDF file named with your student IDs –

**DHW4\_123456789\_300200100.pdf**

1. The submission should contain the following:
   1. The first page should contain the details about the submitters - Name, ID number and email address.
   2. Your answers to the dry part questions.
2. Submission is done electronically via the course website, in the **HW**4 **– Dry** submission box.

# Grading

1. **All** question answers must be supplied with a **full explanation**. Most of the weight of your grade sits on your **explanation** and **evident effort**, and not on the absolute correctness of your answer.
2. Remember – your goal is to communicate. Full credit will be given only to correct solutions which are **clearly** described. Convoluted and obtuse descriptions will receive low marks.

# Questions & Answers

* The Q&A for the exercise will take place at a public forum Piazza **only**. Please **DO NOT** send questions to the private email addresses of the TAs.
* Critical updates about the HW will be published in **pinned** notes in the piazza forum. These notes are mandatory and it is your responsibility to be updated.

A number of guidelines to use the forum:

* Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
* Be polite, remember that course staff does this as a service for the students
* You’re not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
* When posting questions regarding **hw**4, put them in the **hw**4 folder

# Late Days

* Please **DO NOT** send postponement requests to the TA responsible for this assignment. Only the **TA in charge** can authorize postponements. In case you need a postponement, please fill out the attached form: <https://goo.gl/forms/Im4tFXNLq4uznBzM2>

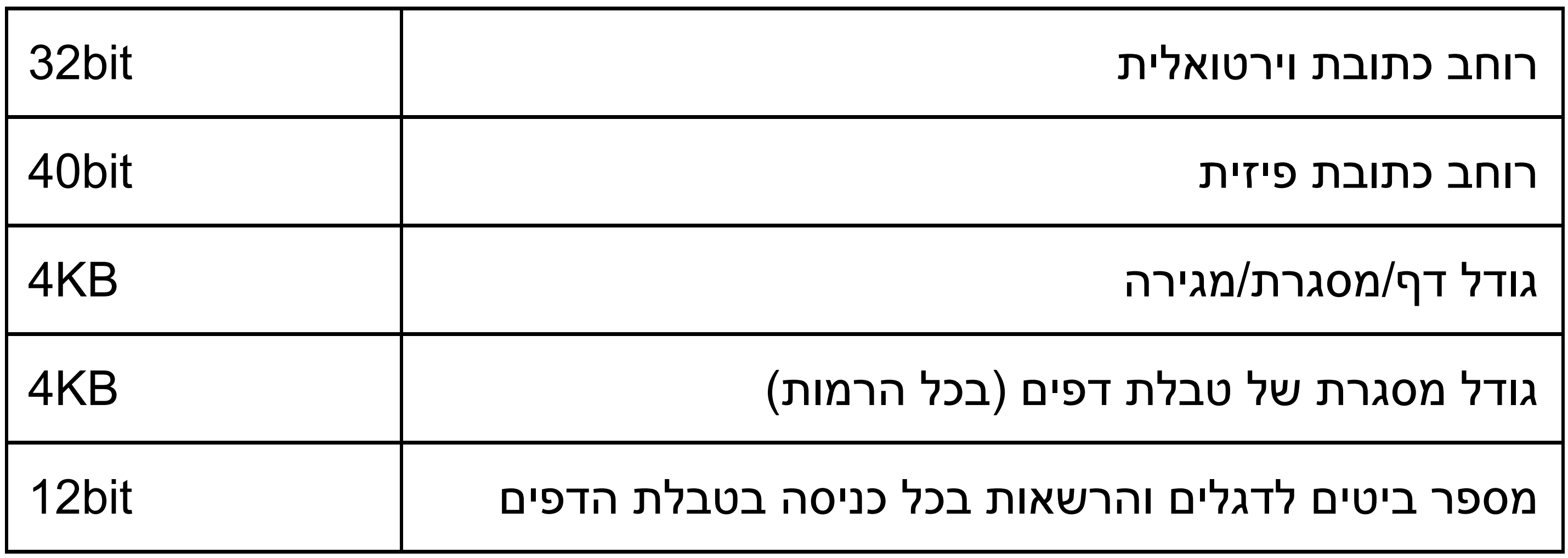
# שאלה 1 - זיכרון:

עדן, זמרת פופולרית, סבלה ממחסור בזיכרון פיזי במחשב שלה (בעל מעבד 32-IA וזיכרון פיזי בגודל 4GB)

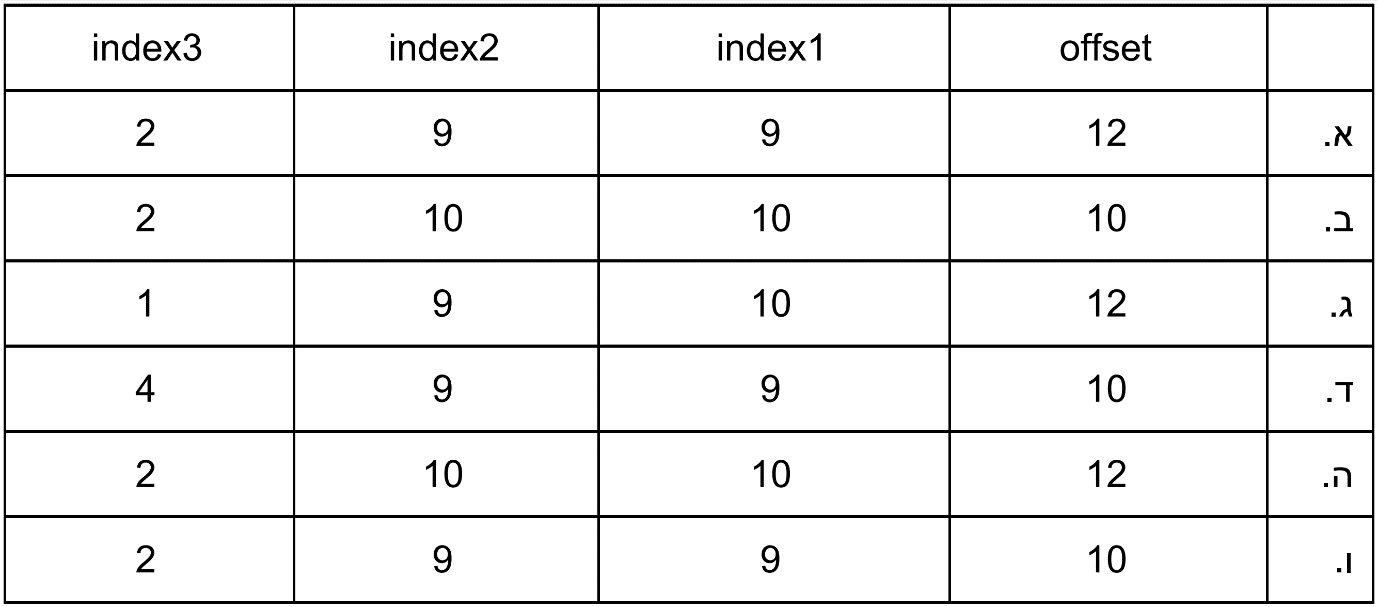
ולכן הציעה תכן חדש של מעבד המרחיב את מרחב הזיכרון הפיזי מ-32 ל-40 ביט. כתוצאה מכך, במימוש של

עדן יש שלוש רמות תרגום בטבלת הדפים.

שאר נתוני המעבד של עדן זהים לאלו של מעבד 32-IA ,כלומר נתוני המערכת החדשה הם:



1. בהנחה שגודל כניסה בטבלת הדפים **מעוגל למעלה לחזקה שלמה של 2**, מהו אופן חלוקת הכתובת הוירטואלית לשדות בתהליך תרגום כתובות? (page walk)



נימוק:

נראה כיצד גודל כניסה בטבלת הדפים צריכה להיראות

גודל מסגרת בזיכרון הוא ולכן זה מחייב את שדה ה- offset להיות 12 סיביות (אחרת לא ניתן לעשות מיפוי לדף שלם בעזרת שדה ה- offset), וזה תואם ל- 12 סיביות המצב שנדרשנו לשמור עליהן.

כמו כן, יש כתובות פיזיות בזיכרון, ולכן מספר המסגרות בזיכרון הפיזי הוא , ולכן נצטרך 28 סיביות בכל כניסה בטבלת הדפים עבור מספר המסגרת הרצוי, לכן גודל כניסה יהיה לפחות 40 סיביות

נצטרך לעגל לחזקה שלמה של 2, ולכן כל כניסה תהיה 64bit המחולקים באופן הבא:

|  |  |  |
| --- | --- | --- |
| State Bits | Frame Number | Sign Extension |
| 12 bits | 28 bits | 64 bits |

כעת אופן חלוקת הכתובות יהיה באופן הבא:

12 סיביות ל- offset מאותם שיקולים שציינו לעיל

גודל כל טבלה הוא 4KB, בעוד שגודל כל שורה הוא 8B, ולכן בכל טבלה יש כניסות, ונצטרך לכל טבלה 9 סיביות. ברמה העליונה נשמור סיביות שנותרו לנו.

ולכן זוהי התשובה הנכונה:



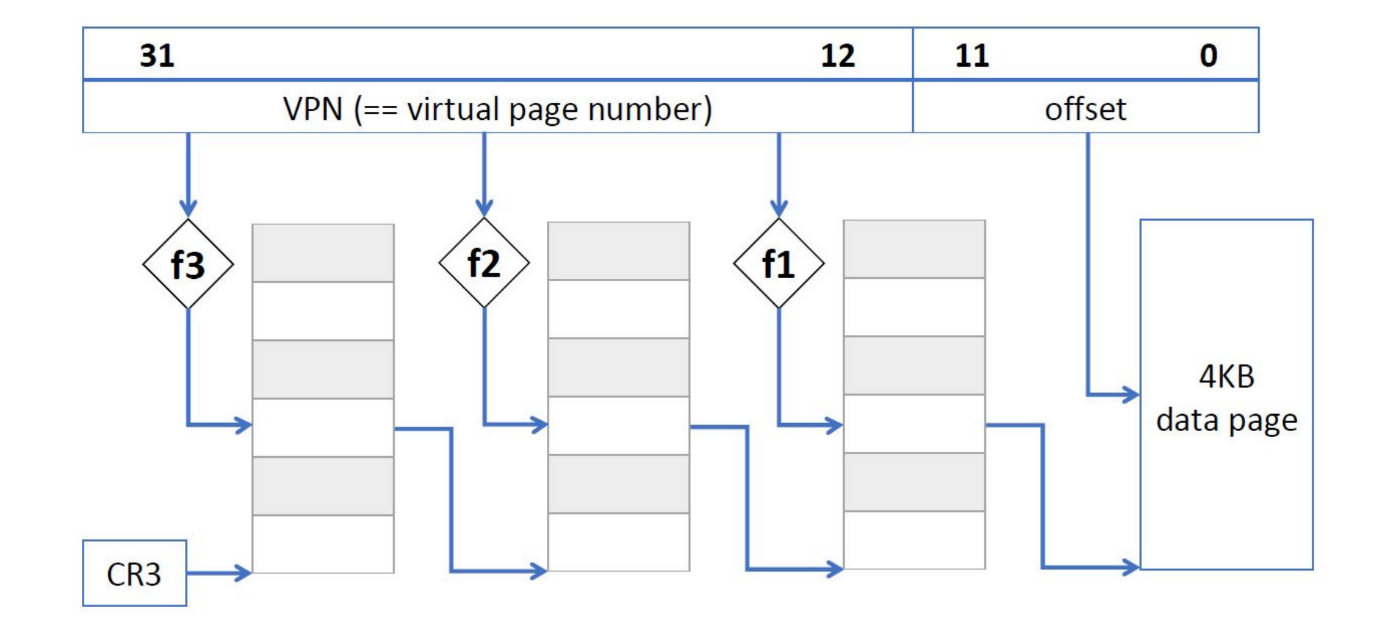
לבעלה של עדן, שוקי, אין שום תואר מהטכניון, ולמרות זאת הוא הבחין כי המימוש של עדן בזבזני בגלל שגודל הכניסות בטבלת הדפים מעוגל למעלה לחזקה שלמה של 2.

1. מהו הגודל המינימלי האפשרי של כניסה בטבלת הדפים אם **לא מעגלים אותו למעלה**?  
   1. 3 בתים
   2. 4 בתים
   3. 5 בתים
   4. 6 בתים
   5. 7 בתים
   6. אף תשובה אינה נכונה

נימוק:

בסעיף א' ראינו כי אנו צריכים בפועל 40 סיביות, כאשר 12 מהן עבור דגלים ו- 28 עבור מספר המסגרת (עשינו את החישוב שבו הגענו לכך שיש לנו בזיכרון הפיזי מסגרות). אם נחלק זאת בגודל בית (8 סיביות) נקבל 5 בתים.

בהמשך לסעיף הקודם, שוקי (בעלה של עדן) הציע מימוש חדש לטבלת הדפים שבו כל כניסה בטבלת הדפים (בכל הרמות) היא בגודל המינימלי מהסעיף הקודם. במימוש של שוקי, כמו במימוש המקורי של טבלת הדפים במעבדי אינטל, דפים סמוכים בזיכרון הוירטואלי נשמרים בכניסות סמוכות בטבלת הדפים. שוקי הבחין שבמימוש החדש הכתובת הוירטואלית אינה מתפרקת לשדות של אינדקסים ויש צורך בחישובים מורכבים על מנת למצוא את האינדקס המתאים בכל טבלה (כלומר בכל רמה בעץ). להלן שרטוט הממחיש את אופן התרגום:



בשרטוט רואים שלוש פונקציות (f1, f2, f3) המקבלות את מספר הדף הוירטואלי VPN ומחזירות, בהתאמה, שלושה אינדקסים לשלושת הרמות בטבלת הדפים. בכל הסעיפים הבאים, הפעולות חלוקה / ומודולו % הן פעולות בשלמים. למשל:

1024/819=1

1024%819=205

1. מהי הפונקציה f1?
   1. f1(vpn) = vpn/819
   2. f1(vpn) = vpn%819
   3. f1(vpn) = (vpn/819)/819
   4. f1(vpn) = (vpn%819)/819
   5. f1(vpn) = (vpn/819)%819
   6. f1(vpn) = ((vpn/819)%819)/819

נימוק:

מתקיים כי בכל טבלה בהצעה של שוקי יש כניסות. במעבד x86-IA32 רגיל נצטרך להסתכל רק על 22 הסיביות העליונות (אליהן נדביק בעזרת bitwise-OR את 12 הסיביות של ה- OFFSET), ודבר זה שקול לביצוע מודולו 1024, כמספר הכניסות בטבלה. במקרה הזה, נצטרך באופן שקול לבצע מודולו 819, כמספר הכניסות בטבלה החדשה.

1. מהי הפונקציה f2?
   1. f1(vpn) = vpn/819
   2. f1(vpn) = vpn%819
   3. f1(vpn) = (vpn/819)/819
   4. f1(vpn) = (vpn%819)/819
   5. f1(vpn) = (vpn/819)%819
   6. f1(vpn) = ((vpn/819)%819)/819

נימוק:

\*\*בתשובות אני מניח שבשאלה התכוונו ל- f2 ולא ל- f1

במעבד רגיל נצטרך רק לחלק ב- 1024 כדי לשלוף את 10 הביטים העליונים. נשים לב כי במעבד של שוקי, אם נחלק ב- 819 לא בהכרח נישאר בטווח הרצוי, ולכן נצטרך לעשות את פעולת המודולו לאחר מכן כדי לקבל תשובה שהיא אכן בטווח בטבלה.

1. מהי הפונקציה f3?
   1. f1(vpn) = vpn/819
   2. f1(vpn) = vpn%819
   3. f1(vpn) = (vpn/819)/819
   4. f1(vpn) = (vpn%819)/819
   5. f1(vpn) = (vpn/819)%819
   6. f1(vpn) = ((vpn/819)%819)/819

נימוק:

נשים לב כי עלינו לקבל את שתי הסיביות הדרושות לבחירת האינדקס השלישי של טבלה בגודל 4. מתקיים כי ומתקיים כי התוצאה תישאר בטווח הייצוג הנתון על ידי 2 הסיביות של אינדקס 3 (ניתן לייצג בעזרת 2 סיביות את המספרים 0 1 2 3) ולכן זאת התשובה הנכונה.

1. מה היתרון של המערכת שהציע שוקי על פני המערכת שהציעה עדן
   1. מיפוי של מרחב זיכרון וירטואלי גדול יותר.
   2. מיפוי של מרחב זיכרון פיזי גדול יותר.
   3. ה-TLB אפקטיבי יותר בגלל שהוא מכסה יותר זיכרון.
   4. טבלאות הדפים של תהליכי משתמש תופסות נפח קטן יותר בזיכרון.
   5. פחות פרגמנטציה חיצונית, כלומר יותר זיכרון רציף.
   6. אף תשובה אינה נכונה.

נימוק:

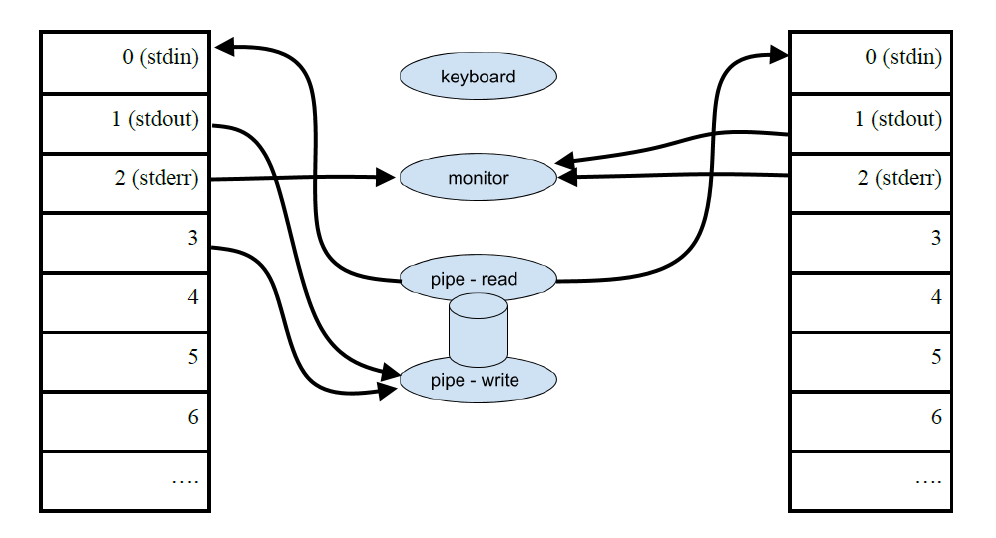
בטבלאות של עדן יש הרבה סיביות לא מנוצלות בכל כניסה, שבהן עשינו SXT לצורך אחידות. בטבלאות של שוקי יש ניצול מיטבי של כל הסיביות הדרושות. לפיכך, בטבלה של שוקי שי 819 כניסות, לעומת טבלה של עדן שבה יש רק 512 כניסות. אם נסתכל על תכנית אשר תשתמש ב-513 כניסות, עדן תצטרך להקצות כבר לפחות 2 טבלאות, אצל שוקי ניתן יהיה להסתפק בטבלה אחת, והמספרים יעלו ככל שהתכנית תצטרך יותר מסגרות בזיכרון.

**שאלה 2 – סיגנלים:**

נתון קטע הקוד הבא:

|  |
| --- |
| void transfer() {// transfer chars from STDIN to STDOUT  char c;  ssize\_t ret = 1;  while ((read(0, &c, 1) > 0) && ret > 0)  ret = write(1, &c, 1);  exit(0); }  int main() {  int my\_pipe[2];  close(0);  printf("Hi");  pipe(my\_pipe);  if (fork() == 0) { // son process  close(my\_pipe[1]);  transfer();  }  close(1);  dup(my\_pipe[1]);  printf("Bye");  return 0; } |

1. השלימו באמצעות חצים את כל ההצבעות החסרות באיור הבא (למשל חץ מ- stdin ל- keyboard), בהינתן שתהליך האב סיים לבצע את שורה 19 ותהליך הבן סיים לבצע את שורה 15:



1. מה יודפס למסך בסיום ריצת שני התהליכים? (הניחו שקריאות המערכת אינן נכשלות):  
   1. Hi
   2. Bye
   3. HiBye
   4. לא יודפס כלום
   5. התהליך לא יסתיים לעולם
   6. לא ניתן לדעת, תלוי בתזמון של התהליכים

נימוק:

בהדפסה הראשונה, לפני הfork, אצל תהליך האב (שהוא היחיד שקיים בזמן זה) stdout עדיין פתוח ומקושר למסך, ולכן ההדפסה הראשונית תתרחש כרגיל.

לאחר ה- fork נקבל את התרשים שעשינו בסעיף 1. כאשר האב יבצע printf, ערוץ הפלט הסטנדרטי שלו יכתוב לצינור, וציר הקריאה יגיע גם ל- stdin של עצמו וגם ל- stdin של תהליך הבן. בעוד שאצל תהליך האב אין טיפול בקלט הזה, אצל תהליך הבן קיימת הפונקצייה transfer אשר תנתב את הקלט שהתקבל מהצינור (וכאמור, pipe-read אצל הבן מעביר את התוכן ל- stdin) אל stdout, ואצל הבן ערוץ זה מקושר למסך ולכן נקבל את ההדפסה HiBye.

בסעיפים הבאים נתבונן בקטע קוד חדש, המשתמש בפונקציה transfer מהסעיף הקודם:

|  |
| --- |
| int my\_pipe[2][2];  void plumber(int fd) {   close(fd);   dup(my\_pipe[1][fd]);   close(my\_pipe[1][0]);   close(my\_pipe[1][1]);   transfer();  }   int main() {   close(0);   printf("Hi");   close(1);   pipe(my\_pipe[0]);   pipe(my\_pipe[1]);    if (fork() == 0) { // son 1   plumber(1);   }   if (fork() == 0) { // son 2   plumber(0);  }   printf("Bye");   return 0;  } |
|  |

1. מה יודפס למסך כאשר תהליך האב יסיים לרוץ? (הניחו שקריאות המערכת אינן נכשלות) רמז: שרטטו דיאגרמה של טבלאות הקבצים כפי שראיתם בסעיף 1.

Hi .a

Bye .b

HiBye .c

ByeHi .d

e. לא יודפס כלום

f. לא ניתן לדעת, תלוי בתזמון של התהליכים...

נימוק:

בהתחלה תהליך האב ידפיס hi בלי תלות בדבר, שכן STDOUT שלו עדיין מצביע למסך. לאחר מכן, שני הצינורות יצונרו אצל תהליך האב ויועברו גם לבנים, ובכך ועקב הסגירות אצל תהליך האב הפלט הסטנדרטי ינותב לכיוון **כתיבה** ב- pipe[0], ואז תהליך האב ייגמר.

לאחר מכן, עקב פעולות ה- plumber אשר יתרחשו, נקבל מצב כי בו יש לנו שני צינורות, כאשר תהליך בן 1 מקבל את stdin מ- pipe-read של צינור א' וכותב בעזרת stdout ל- pipe-write של צינור ב', ותהליך ב' יעשה את הפעולה ההפוכה. מדובר בקריאות מערכת חוסמות, ולכן הבנים יישארו בלולאה אינסופית של קריאה וכתיבה מהצינורות ללא תנאי עצירה ולא ידפיסו דבר.

אפולו, אל האור, שמע שסטודנטים רבים בקורס עובדים עכשיו על תרגיל הבית במקום ליהנות בחוץ בשמש. בתגובה נזעמת, אפולו הכל יכול התחבר לשרת הפקולטה והריץ את התוכנית הנ"ל N פעמים באופן סדרתי (דוגמה ב-bash ,כאשר a.out הוא קובץ ההרצה של התכנית הנ"ל):

>> for i in {1..N}; do ./a.out;

1. אחרי שהלולאה הסתיימה, נשארו במערכת 0 או יותר תהליכים חדשים.

מה המספר המינימלי של סיגנלים שצריך לשלוח באמצעות kill על מנת להרוג את כל התהליכים החדשים שסנטה יצר?

* 1. 0
  2. 1
  3. N
  4. N/2
  5. 2N
  6. לא ניתן לדעת, תלוי בתזמון של התהליכים…

נימוק:

אפולו יוצר לנו בלולאה בטרמינל שלו N תהליכים, וכאמור יהיו לנו 2N זוגות של תהליכים בלולאות אינסופיות אשר ממתינים זה לזה. אם יישלחו N סיגנלים כאשר כל סיגנל יישלח לזוג אחר, נקבל את המצב הבא:

תהליך א' קרא מהצינור וכתב לשם בחזרה, התהליך השני לא שם כדי לקרוא, ולכן התהליך הראשון יקרא שוב, ייקרא EOF ולכן ישתחרר מהלולאה, ובכך יגיע ל- return 0 ב-main ויסיים את ריצתו.

1. מה תהיה התשובה עבור הסעיף הקודם אם נסיר את שורות 5-6 מהקוד?
2. 0
3. 1
4. N
5. N/2
6. 2N
7. לא ניתן לדעת, תלוי בתזמון של התהליכים…

נימוק:

כעת לא סוגרים את ה-FD של בן 2 שמצביע ל- pipe[1] read ולכן כאשר בן 2 ינסה לקרוא מתוך הצינור הנ"ל הוא לא יקבל EOF (כי כביכול עדיין יש תהליכים שכותבים אליו) אלא ישלח להמתנה עד שיכתב משהו לתוךpipe[1] write (אבל למעשה זהו אותו בן 2 עצמו שמצביע לצינור הנ"ל ולכן גם לעולם לא יכתב לשם כלום).

לכן אם נשלח SIGKILL ל-son1 אנו נכנס לסיטואציה שתיארנו לעיל (בן 2 ממתין לנצח על צינור 1 לקלט שלא יגיע מאחר ואין באמת כותבים לpipe[1] write אלא פשוט f\_count של אותו ה-FD לא מאופס מאחר ובן 2 עדין מצביע עליו מתוך אינקס: FDT[4] ) . זהו מצב של deadlock .

לעומת זאת אם נשלח SIGKILL ל-son2 לא ניכנס לdeadlock מאחר ובן 1 קורא מ-pipe[0] read ול- pipe[0] write מצביע רק בן 2 . לכן כשימות בן 2 יאופס מונה המצביעים של ה-FO של pipe[0] write וכשבן 1 ינסה לקרוא ממנו הוא יקבל EOF ולא ימשיך את ההמתנה אלא יסיים את ריצתו.

לכן ניתן לשלוח מספר מינימלי של N סיגנלים , אלא שכעת יש אילוץ לשלוח אותם רק לבנים מסוג son2.