

Submitters:

Tom Guy

315155671

[tom.guy@campus.technion.ac.il](mailto:tom.guy@campus.technion.ac.il)

Nikita Ivanov

322245101

[Nikitaiv@campus.technion.ac.il](mailto:Nikitaiv@campus.technion.ac.il)

Operating Systems – 234123

Homework Exercise 1 – Dry

Spring 2022

Teaching assistant in charge:

Ayala Fostick-Tur

Assignment Subjects & Relevant Course material

Processes and inter-process communications

Recitations 1-3 & Lectures 1-3

Page 1 of 9

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 –

DHW1\_123456789\_300200100.pdf

3. The submission should contain the following:

a. The first page should contain the details about the submitters - Name, ID number, and

email address.

b. Your answers to the dry part questions.

4. Submission is done electronically via the course website, in the HW1 – 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 hw1 , put them in the hw1 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://forms.office.com/r/6AELMPGcGY

Page 2 of 9

Question 1 – Process management (50 points)

Part 1

int X = 1, p1 = 0, p2 = 0;

int ProcessA() {

printf("process A\n");

while(X);

printf("process A finished\n");

exit (1);

}

void killAll(){

if(p2) kill(p2, 15);

if(p1) kill(p1, 9);

}

int ProcessB() {

X = 0;

printf("process B\n");

killAll();

printf("process B finished\n");

return 1;

}

int main(){

int status;

if((p1 = fork()) != 0)

if((p2 = fork()) != 0){

wait(&status);

printf("status: %d\n", status);

wait(&status);

printf("status: %d\n", status);

} else {

ProcessB();

} else {

ProcessA();

}

printf("The end\n");

return 3;

Page 3 of 9

בשאלה זו עליכן להניח כי :

. 1 קריאות המערכת fork () ו kill () אינן נכשלות.

. 2 כל שורה הנכתבת לפלט אינה נקטעת ע"י שורה אחרת.

. 3 כאשר תהליך מקבל סיגנל x הוא מסתיים וערך היציאה שלו הוא x + 128 .

עבור כל אחת משורות הפלט הבאות, סמנו כמה פעמים הן מופיעות בפלט כלשהו, נמקו את תשובתכן.

process A . 1

0 . a

0 or 1 . b

1 . c

1 or 2 . d

2 . e

נימוק: התהליך הנוצר בפיצול הראשון (הבן הנוצר ב-fork()) הוא זה שיכנס לתוך ProcessA() אך אין אפשרות לדעת מתי. במידה והתהליך הנוצר בפיצול השני ונכנס לתוך ProcessB() יופעל קודם ע"י מערכת ההפעלה ויקרא ל-kill(p1,9) (כאשר p1 מחזיק את ה-pid של התהליך שנוצר בפיצול הראשון) לפני הדפסת Process A, בעת שמערכת ההפעלה תפעיל את התהליך שה-pid שלו הוא p1, שגרת הטיפול בסיגנלים תטפל בסיגנל שנשלח ולא יודפס כלל Process A. באותו אופן, יכול להיות שהשורה kill(p1,9) תיקרא לאחר ההדפסה. לכן יתכן ששורה זו תודפס פעם אחת או לא בכלל.

status: 1 . 2

0 . a

0 or 1 . b

1 . c

1 or 2 . d

2 . e

נימוק: המקום היחיד בו שורה זו יכולה להיות מודפסת בקטע הקוד הנתון הוא בשורות לאחר הקריאות ל-wait. הערך הנמצא ב-status הוא ערך ההחזרה של סיום תהליך הבן שיסתיים ואותה זיהתה פעולת ה-wait. הערך 1 שמור לפעולה init ולא ייתכן שהיא תסתיים במהלך ריצת התוכנית ולכן שורה זו לא תודפס כלל.

status: 137 . 3

0 . a

0 or 1 . b

1 . c

1 or 2 . d

2 . e

נימוק: תהליך A לא יכול להסתיים לפני הקריאה ל-kill ולהחזיר ערך שונה ולכן ערך ההחזרה שלו בסיום התוכנית חייב להיות 137. כמובן שרק תהליך אחד יסתיים בקריאה ל-kill(p1,9) ולכן שורה זאת תודפס פעם אחת בלבד.

Page 4 of 9

לחךלחךל

status: 143 . 4

0 . a

0 or 1 . b

1 . c

1 or 2 . d

2 . e

נימוק: תהליך B חייב להסתיים בסיום הריצה שלו ולא בקריאה ל-kill מפני שבעת ריצת תהליך שהמזהה שלו p2, בקריאה ל-kill ישלח 0 כי זה הערך שיוחזר אחרי ה-fork (כלומר, בעצם יקרה kill(p2,15)=kill(0,15)). לכן שום תהליך לא יסתיים מהסיגנל 15 ולא יוחזר ל-status 143.

בסיום

The end . 5

0 . a

0 or 1 . b

1 . c

1 or 2 . d

2 . e

נימוק: התהליך המקורי והתהליך שנוצר מה-fork השני, שניהם ורק הם יגיעו לשורה זו בסוף ריצתם.

Part 2

עבור אותו קוד כמו בחלק הקודם ­ סמנו עבור כל שורה האם היא יכולה להתקיים בפלט כלשהו, ונמקו.

. 1 השורה process B מופיעה לפני השורה process A . אפשרי / לא אפשרי

נימוק: מערכת ההפעלה מחליטה את איזה תהליך להריץ קודם. יתכן בהחלט שהיא תחליט להמשיך להריץ רת תהליך האב אחרי ה-fork הראשון ואחרי השני להריץ את התהליך שנוצר ובכך אכן יודפס process B לפני process A.

. 2 השורה 137 : status מופיעה לפני השורה The end . אפשרי / לא אפשרי

נימוק: ­אם שורה status: 137 תודפס, אז היא חייבת להיות מודפסת קודם.

. 3 השורה 143 : status מופיעה לפני השורה .137 : status אפשרי / לא אפשרי

נימוק: כמו שהוסבר קודם, השורה status: 143 לא תודפס כלל.

Page 5 of 9

Question 2 – Inter-Process Communication (50 points)

נתון קטע הקוד הבא:

1. void transfer() {// transfer chars from STDIN to STDOUT

2. char c;

3. ssize\_t ret = 1;

4. while ((read(0, &c, 1) > 0) && ret > 0)

5.

ret = write(1, &c, 1);

6. exit(0);

7. }

8.

9. int main() {

10. int my\_pipe[2];

11. close(0);

12. printf("Hi");

13. pipe(my\_pipe);

14. if (fork() == 0) {

15.

close(my\_pipe[1]);

16.

transfer();

17. }

18. close(1);

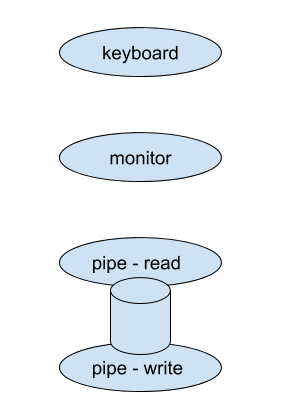
19. dup(my\_pipe[1]);

20. printf("Bye");

21. return 0;

22. }

Page 6 of 9



. 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

|  |
| --- |
| 0 (stdin) |
| 1 (stdout) |
| 2 (stderr) |
| 3 |
| 4 |
| 5 |
| 6 |
| …. |

|  |
| --- |
| 0 (stdin) |
| 1 (stdout) |
| 2 (stderr) |
| 3 |
| 4 |
| 5 |
| 6 |
| …. |

. 2 מה יודפס למסך בסיום ריצת שני התהליכים? (הניחו שקריאות המערכת אינן נכשלות ):

a . Hi

b . Bye

c . HiBye

d . לא יודפס כלום

e . התהליך לא יסתיים לעולם

f . לא ניתן לדעת, תלוי בתזמון של התהליכים

נימוק:

ראשית, בשורה 11 מתבצעת סגירה לערוץ הקלט ולכן אף לאף אחד מהתהליכים לא תהיה אפשרות לקבל קלט מהערוץ הסטנדרטי (נובע בין היתר מכך שלתהליך הבן יש טבלה המועתקת מזו של תהליך האב ולכן בעת יצירתו גם לו ערוץ הקלט יהיה סגור). שנית, בשורה 12 מודפסת השורה - ולכן היא בהכרח תודפס למסך. מעבר לכך, תהליך הבן ידפיס בפונקציה - את הקלט הוא יקבל דרך הצינור (אינדקס 0 בטבלת הקבצים של הבן, כמו של האב, מופנה לחלק הקריאה מהצינור בעקבות סגירת ערוץ הקלט הסטנדרטי ופתיחת הצינור לאחר מכן) וידפיס למסך דרך ערוץ הפלט הסטנדרטי שלא נסגר אצלו.

transfer()

Hi

Bye

**\*לוודא שקריאת המערכת (שורה 4) מחכה לקלט עד שהוא מגיע. אחרת ייתכן והמילה השנייה לא תודפס והתשובה הנכונה היא האחרונה.**

Page 7 of 9

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

1. int my\_pipe[2][2];

2. void plumber(int fd) {

3. close(fd);

4. dup(my\_pipe[1][fd]);

5. close(my\_pipe[1][0]);

6. close(my\_pipe[1][1]);

7. transfer();

8. }

9.

10. int main() {

11. close(0);

12. printf("Hi");

13. close(1);

14. pipe(my\_pipe[0]);

15. pipe(my\_pipe[1]);

16.

17 if (fork() == 0 ) {

plumber(1);

18.

19. }

20. if (fork() == 0) { // son 2

plumber(0);

21.

22. }

23. printf("Bye");

24. return 0;

25. }

. 3 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: בשורה 12 יודפס למסך Hi ולאחר מכן מה שיקרה בעקבות ה-pipe-ים והפונקציות dup, close הוא שערוץ הפלט של תהליך האב יהיה לכתיבה לצינור הראשון, ממנו קורא הבן הראשון, הבן הראשון כותב לצינור השני וממנו קורא הבן השני שכותב לצינור הראשון. לכן בעצם שני הבנים יכנסו ללולאה אינסופית **(שוב אותה הערה כמו קודם)** ועד סוף תהליך האב לא יודפס דבר נוסף למסך.

Page 8 of 9

נועה קירל שמעה שסטודנטים רבים בקורס עבדו במהלך יום הסטודנט על תרגיל הבית, ואפילו נהנו ממנו יותר

מאשר בפסטיבל של הטכניון. בתגובה נזעמת, נועה התחברה לשרת הפקולטה והריצה את התוכנית הנ"ל N פעמים

באופן סדרתי (דוגמה ב­ bash ,כאשר out.a הוא קובץ ההרצה של התכנית הנ"ל):

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

. 4 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 שנועה יצרה?

0 . a

1 . b

c . N

N/2 . d

e . 2N

f . לא ניתן לדעת, תלוי בתזמון של התהליכים …

נימוק: שני התהליכים שנוצרו מכל הרצה (ב-fork-ים) לא יסתיימו ללא קריאה ל- kill אך כל קריאה ל-kill עבור אחד מהזוגות תגרום למצב שלבן השני אין מי שיכתוב אליו\יקרא ממה שהוא כותב ולכן הוא ייסגר לבד. מסקנה: מספיק N קריאות.

. 5 מה תהיה התשובה עבור הסעיף הקודם אם נסיר את שורות 5­6 מהקוד ?

0 . a

1 . b

c . N

N/2 . d

e . 2N

f . לא ניתן לדעת, תלוי בתזמון של התהליכים…

נימוק: מקרה זה אינו סימטרי לראשון, כי לא מספיק לסגור את הבן השני, כי אז הבן הראשון יכנס ללולאה אינסופית בה הוא כותב וקורא מעצמו, אך סגירת הבן הראשון תביא למצב שבו הבן השני ייסגר גם כן כי לא יהיה מי שיכתוב לו או יקרא מה שהוא כותב. לכן מספיקות N קריאות ל-kill.

בהצלחה !

Page 9 of 9

|  |
| --- |
| 0 (stdin) [0][0] |
| 1 (stdout)[0][1] |
| 2 (stderr) |
| 3 [1][0] |
| 4 [1][1] |
| 5 |
| 6 |
| …. |

|  |
| --- |
| 0 (stdin) [0][0] |
| 1 (stdout)[0][1] |
| 2 (stderr) |
| 3 [1][0] |
| 4 [1][1] |
| 5 |
| 6 |
| …. |

Diagram

Description automatically generated

Diagram

Description automatically generated

|  |
| --- |
| 0 (stdin) [0][0] |
| 1 (stdout)[0][1] |
| 2 (stderr) |
| 3 [1][0] |
| 4 [1][1] |
| 5 |
| 6 |
| …. |