TECHNION
Israel Institute of Technology

Computer Science Department

Operating Systems – 234123

Homework Exercise 1 – Dry

Winter 2024

Hillah Hassan 209583574 hillahhassan@campus.technion.ac.il

Daniel Tarko 941200610 tarko@campus.technion.ac.il

Question 1 – Inter-Process Communication (45 points)

שאלה זו מתייחסת למערכת ההפעלה לינוקס.

חלק א:

נתון תהליך רץ A.

1. (8 נק') האם *יתכן* שהאירועים הבאים יגרמו להחלפת הקשר מיידית מ-A? אם כן, תנו דוגמא. אחרת, נמקו.

אירוע	כן / לא	דוגמא / נימוק
פסיקת שעון (timer interrupt)	כן	מערכת ההפעלה יכולה להחליט שתהליך רץ יותר מידי זמן ולעשות החלפת הקשר מיידית לתהליך אחר. ראינו זאת בscheduling.
תהליך A מבצע syscall	כן	קריאת מערכת חוסמת למשל wait() תגרום לאבא לחכות לבן שלה. כלומר יתכן והיא תבצע החלפת הקשר אל הבן אם עוד לא סיים.

2. (8 נק') האם האירועים הבאים יגרמו *בהכרח* להחלפת הקשר מיידית מ-A? נמקו.

אירוע	כן / לא	נימוק
פסיקת שעון (timer interrupt)	לא	יתכן ועוד לא עבר פרק הזמן שהוקצב עבור תהליך.
תהליך A מבצע syscall	לא	למשל הקריאת מערכת getpid אינה גורמת להחלפת הקשר

3. (15 נק') האם האירועים הבאים יגרמו *בהכרח* למעבר מיידי ממצב משתמש ב-A למצב גרעין? נמקו.

אירוע	כן / לא	נימוק
פסיקת שעון (timer interrupt)	כן	פסיקה תמיד גורמת מעבר למצב גרעין. נציין גם שהטיפול בה הוא במצב גרעין.
תהליך A מבצע syscall	כן	קריאות מערכת מעבירות את השליטה למערכת ההפעלה.
תהליך שרץ על ליבה אחרת שולח SIGINT לתהליך A	לא	הסיגנל יכתב במערך הסיגנלים, אך המערך יבדק רק כאשר התהליך חוזר ממצב גרעין.

4. (4 נק') האם fork יוצרת מופע חדש בעבור האובייקטים הבאים:

	מחסנית	File objects	FDT	PCB
כן / לא	כן	לא	כן	כן

חלק ב: (10 נקודות)

לפניכם קטע קוד:

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

void fpe_catcher (int signum) {
  printf("Hello\n");
  exit(0);
}
int main() {
  signal(SIGFPE, fpe_catcher);
  int x = 234123 / (0);
  printf("Hi\n");
  while(1);
  return 0;
}
```

תזכורת:

SIGFPE הוא הסיגנל המתאים לשגיאות אריתמטיות, כמו חלוקה ב-0 (גם במקרה של מספרים שלמים).

1. בחרו באפשרות הנכונה בנוגע לריצת הקוד: (4 נקודות)
1. יודפס קודם "Hi" ואז "Hello".
2. יודפס רק "Hello".
3. יודפס רק "Hi".
4. לא יודפס כלום.
5. תשובות i,ii אפשריות.

נמקו:

בתחילת ריצת התכנית, מתבצעת הקריאת מערכת signal אשר במקרה זה מגדירה שטיפול סיגנל מסוג SIGFPE יהיה על ידי הפונקציה fpe_catcher. בהנחה וקריאת המערכת מצליחה נגיע לשורה הבאה אשר מערבת חלוקה באפס. המעבד יצור interrupt וכתוצאה מכך מערכת ההפעלה תשלח את הסיגנל SIGFPE לתהליך האחראי. התהליך יקרא לhandler החדש שלנו. ב handler החדש מודפס Hello והתהליך מסתיים בעקבות exit .ובפרט התכנית לא ממשיכה לשורות הבאות.

2. כעת נניח שבזמן כלשהו של ריצת הקוד הנ"ל, תהליך אחר שולח את הסיגנל SIGFPE לתהליך המריץ את הקוד.

עבור כל אחד מהתרחישים הבאים, הכריעו האם הוא **אפשרי** או לא, ונמקו בקצרה: (4=x22 נקודות)

- יודפס "Hello" פעמיים.

הקיפו: כן \ לא

נימוק:

על מנת שHello יודפס פעמיים צריך שהhandler ירוץ פעמיים. הדבר אינו אפשרי מכיוון שיש קריאה לexit בסוף הhandler שיסיים את התהליך (אז בעצם התהליך מת לפני שהוא יכול להמשיך לטפל בסיגנל השני). ואולי אפשרי שהסיגנל מהתהליך השני מגיע באמצע הטיפול של הסיגנל הראשון ואז רגע לפני הexit עבור הטיפול בתהליך הראשון יודפס Hello מהטיפול של התהליך השני? אז גם זה לא! בעת טיפול בסיגנל מסוג x מערכת ההפעלה חוסמת טיפול בסיגנלים נוספים מסוג x על מנת למנוע reentrancy.

- לא יודפס "Hello" בכלל.

הקיפו: כן \ לא

נימוק:

יתכן ונשלח סיגנל מהתהליך השני לפני ביצוע השורה הראשונה בmain. במצב כזה יקרא הhandler הדיפולטיבי של SIGPFE שפשוט הורג את התהליך.

3. כעת הניחו שוב כי לא נשלח סיגנל לתהליך המריץ את הקוד ע"י תהליך אחר. תארו את ריצת התכנית במידה והיינו מסירים את פקודת ה-exit(0): (2 נקודות)

הקוד יכנס ללואה אינסופית שמדפיסה "Hello" כי הhandler לא עושה שום דבר לטפל בחלוקה ב0. אז כשאר השורה של החלוקה תרוץ שוב (לאחר הקריאה לhandler) שוב יקרא הhandler וידפיס Hello וכן הלאה אינסוף פעמים.

Question 2 – Process management (40 points)

```
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;
     }
```

בשאלה זו עליכן להניח כי:

1. קריאות המערכת fork() וkill() 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.

1. process A
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק: אופציה אחת הבן השני (שני לפי סדר יצירה) יריץ את processB() לפני שהבן הראשון יריץ את ProcessA. במצב זה processB() יהרוג את תהליך הבן הראשון. ככה הבן הראשון לא יספיק להריץ את processA() ובפרט לא יודפס "process A". אופציה נוספת זה שדווקא הבן הראשון מריץ קודם את processA(), מדפיס "process A" מתחיל את הלולאה האינסופית שלו ורק אז הבן השני הורג את התהליך הראשון.

2. status: 1
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק: בעיקרון, רק התהליך בן הראשון מסוגל להחזיר status 1 אבל זה לעולם לא יקרה כי הוא יכנס ללואה אינסופית וישלח אליו סיגנל 9 מהבן השני (SIGKILL) ואז יודפס status: 137 (כאשר 137= 128 + 9).
כמו כן, תהליך הבן השני לא יגרום להדפסת status 1 למרות שprocessB() מחזיר 1.
הסיבה לכך היא שתהליך הבן השני ממשיך עד סוף המain ששם מוחזר הערך 3. לכן עבורו יודפס status: 3.

3. status: 137
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק:

כפי שהסברנו קודם, תהליך הבן הראשון יתקע בלולאה אינסופית על שתהליך הבן השני ישלח לבן הראשון SIGKILL. זהו סיגנל מספר 9. ואז תהליך האבא ידפיס status:137 (כאשר 137= 128 + 9).

4. status: 143

a. 0
b. 0 or 1
c. 1
d. 1 or 2
e. 2

נימוק: הדרך לקבל הדפסה של 143 היא אם נשלח לתהליך בן כלשהו הסיגנל 143-128 = 15. רק בפונקציה killAll מופיע שליחת סיגנל 15 לתהליך הבן השני. אך נשים לב כי שליחת הסיגנל מותנית בערכו של p2. מכיוון שתהליך הבן השני הוא זה שמריץ פונקציה זו, הערך של p2 הוא בהכרח 0. ולכן לא תהיה כניסה לif ולא ישלח סיגנל 15 לתהליך הבן השני.

5. The end

a. 0
b. 0 or 1
c. 1
d. 1 or 2
e. 2

נימוק: רק שתי תהליכים יגיעו לשורת הדפסה printf("The end"). תהליך האבא ותהליך הבן השני. (תהליך הבן הראשון יהרג על ידי הבן השני).