מערכות הפעלה

תרגול 13 –פתרון שאלות

מתרגל-יורם סגל
yoramse@colman.ac.il

שני אלקובי
פריאל לוי

Contents

1 חזרה למבחן מספר 3

Mutex vs Semaphores

נושא	Semaphores	Mutex
מנעול בינארי	כן	כן
הרשאת גישה לקטע קוד קיריטי ליותר מחוט אחד בו זמנית	כן	לא
בעלות	**אין בעלות, כל חוט יכול לשנות כל מנעול**	**רק מי שנעל יכול לשחרר**

לסיכום:

MUTEX עדיף (ואפשרי) רק במקרה של מנעול בינארי

חוטים

- תהליך ב-Linux יכול לכלול מספר חוטים המשתפים ביניהם את כל משאבי התהליך :
  - מרחב הזיכרון.
  - גישה לקבצים והתקני חומרה.
  - מנגנונים שונים של מערכת ההפעלה.

כל חוט בתהליך מהווה הקשר ביצוע נפרד – לכל חוט מחסנית ורגיסטרים משלו.

תהליכים בהשוואה לחוטים

נושא	תהליכים	חוטים
ביצועים	כבד	קל
זמני הפעלה וכיבוי	איטי	מהיר
תיקשורת פנימית	מורכבת	פשוטה (שיתוף)
מרחב כתובות	פועל במרחב סגור	פתוח
השפעת קריסה	אין השפעה על אחרים	לוקח איתו את החוטים האחרים של אותו התהליך
אבטחת מידע (גלישה)	מאובטח	פירצת אבטחת מידע

לסיכום:

תהליכים עדיפים במקרה שדרוש חסינות לבגים ואבטחת מידע תוך התפשרות על זמני התחלה וסיום.

חוטים עדיפים כאשר ביצועים הם הדבר החשוב ביותר ואין בעית שיתוף מידע.

חוטים

- pthread_join(pthread_t th, void **thread_return)
- pthread_cancel(pthread_t thread)
- pthread_exit(void *retval)

סיום חוט

- חוט יכול להסתיים כתוצאה ממספר אפשרויות שונות:
  - חזרה מהפונקציה הראשית של החוט.
  - קריאה ל-pthread_exit() בתוך קוד החוט.
  - קריאה ל-exit() ע"י חוט כלשהו בקבוצה של החוט המדובר.
  - סיום "טבעי" של החוט הראשי.
  - הריגת החוט ע"י קריאה ל-pthread_cancel() מחוט אחר כלשהו ביישום.

הגנה על קטע קריטי

- pthread_mutex_init(pthread_mutex_t **mutex*, const pthread_mutexattr_t **attr*)

attr :

- **PTHREAD_MUTEX_NORMAL** - for “fast” mutexes
- **PTHREAD_MUTEX_RECURSIVE** - for “recursive” mutexes.
- **PTHREAD_MUTEX_ERRORCHECK** - for “error checking” mutexes.
- **PTHREAD_MUTEX_DEFAULT**

הגנה על קטע קריטי

- pthread_mutex_destroy(pthread_mutex_t *mutex)
- pthread_mutex_lock(pthread_mutex_t *mutex)
- pthread_mutex_trylock(pthread_mutex_t *mutex)
- pthread_mutex_unlock(pthread_mutex_t *mutex)

שאלה 1

```
pthread_t ntid[2];
void* thr_fn(void *arg) {
int i;
for(i=0;i<3;i++)
  sleep(1);
  printf("my tid is %u, my pid is: %d\n”,
pthread_self(),    getpid());
  return((void *)0);
}

int main(void) {
  int status, i;
  for(i=0;i<2;i++) {
    status = pthread_create(&ntid[i], NULL, thr_fn,
NULL);
    if (status != 0)
    perror("can't create thread\n");
  }
```

הציעו שני פתרונות על מנת לוודא שכל החוטים יסיימו את עבודתם ואכן יודפסו למסך ההדפסות הרצויות במלואן לפני שהתהליך יסתיים (אין להשתמש ב-sleep).

שאלה 2

```
#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include<stdlib.h>


int main (void){
        printf("waiting for SIGCHLD\n");
        pause();
        printf("waiting for SIGCHLD again\n");
        pause();
        printf("I'm done\n");
        return 0;
}
```

א. הניחו שקיים תהליך כלשהו השולח ברצף סיגנלים מסוג SIGCHLD(שפעולת ברירת המחדל שלו היא ignore) לתהליך המריץ את התוכנית הנתונה, מה יודפס על המסך כתוצאה מריצת התוכנית?

תשובה

א. waiting for SIGCHLD

הסבר: לאחר ההדפסה הראשונה , התהליך נכנס למצב "הפסקה" עד שיקבל איתות כלשהו שהוגדר עבורו טיפול על ידי התהליך- רק במקרה זה יתעורר ממצב ה"הפסקה" וימשיך לרוץ. כלומר, כיוון שהתהליך לא הגדיר טיפול באיתות הSIGCHLD הוא לא יתעורר עקב הגעת איתות זה.

שאלה 3

```
int main()
{
        int status;
        printf("1\n");
        fork();
        wait(&status);
        printf("2\n");
        fork();
        wait(&status);
        printf("3\n");
}
```

כמה פלטים שונים אפשריים עבור הרצת התוכנית הבאה (בהנחה שהתוכנית רצה ללא שגיאות)?

שאלה 3

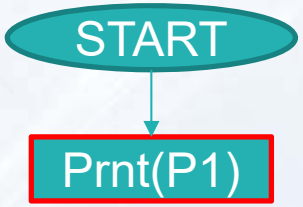
START

```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
```

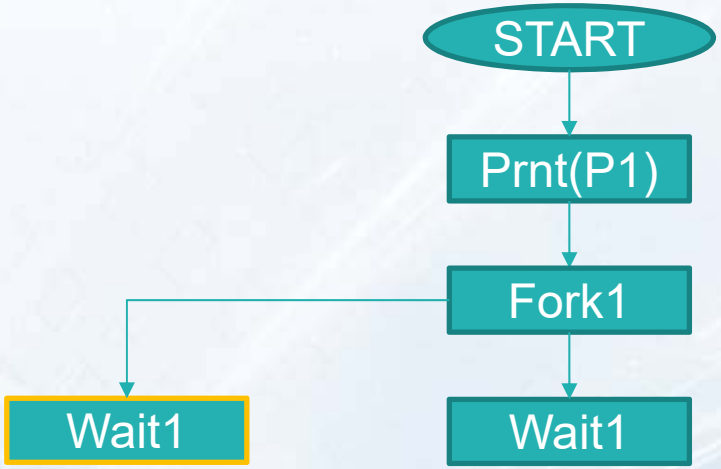



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
```

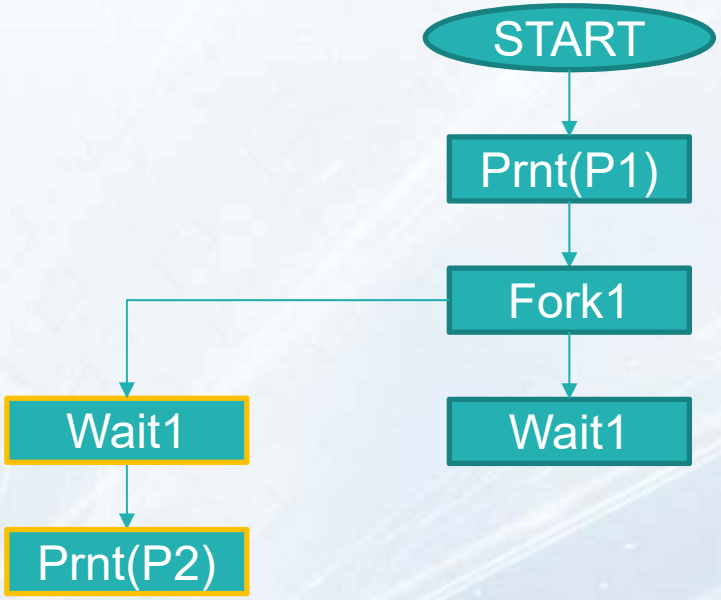



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

2

שאלה 3

```
before any fork 33591
After first fork 33592
```



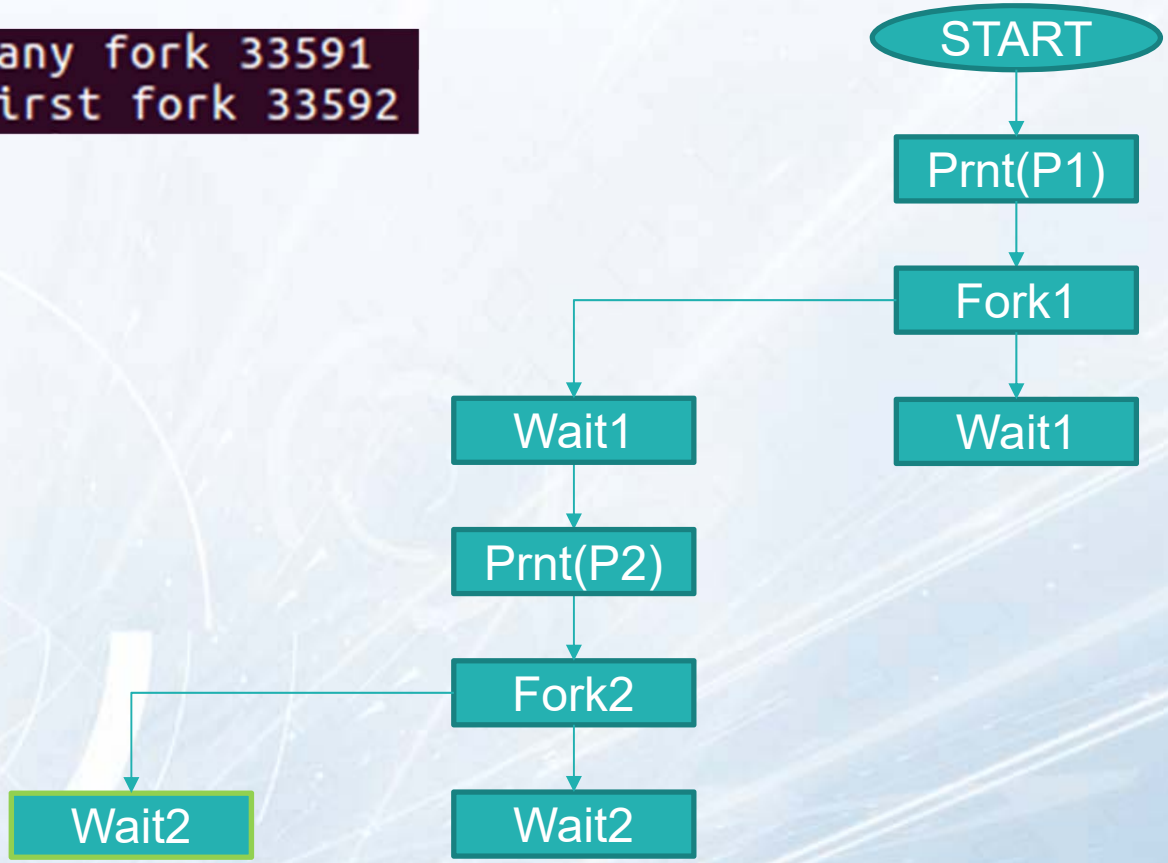

```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

2

נשים לב: לבן אין ילד ולכן
אין משמעות לWAIT שלו.
והוא ממשיך כרגיל.
האב ממתין!!!

שאלה 3

```
before any fork 33591
After first fork 33592
```

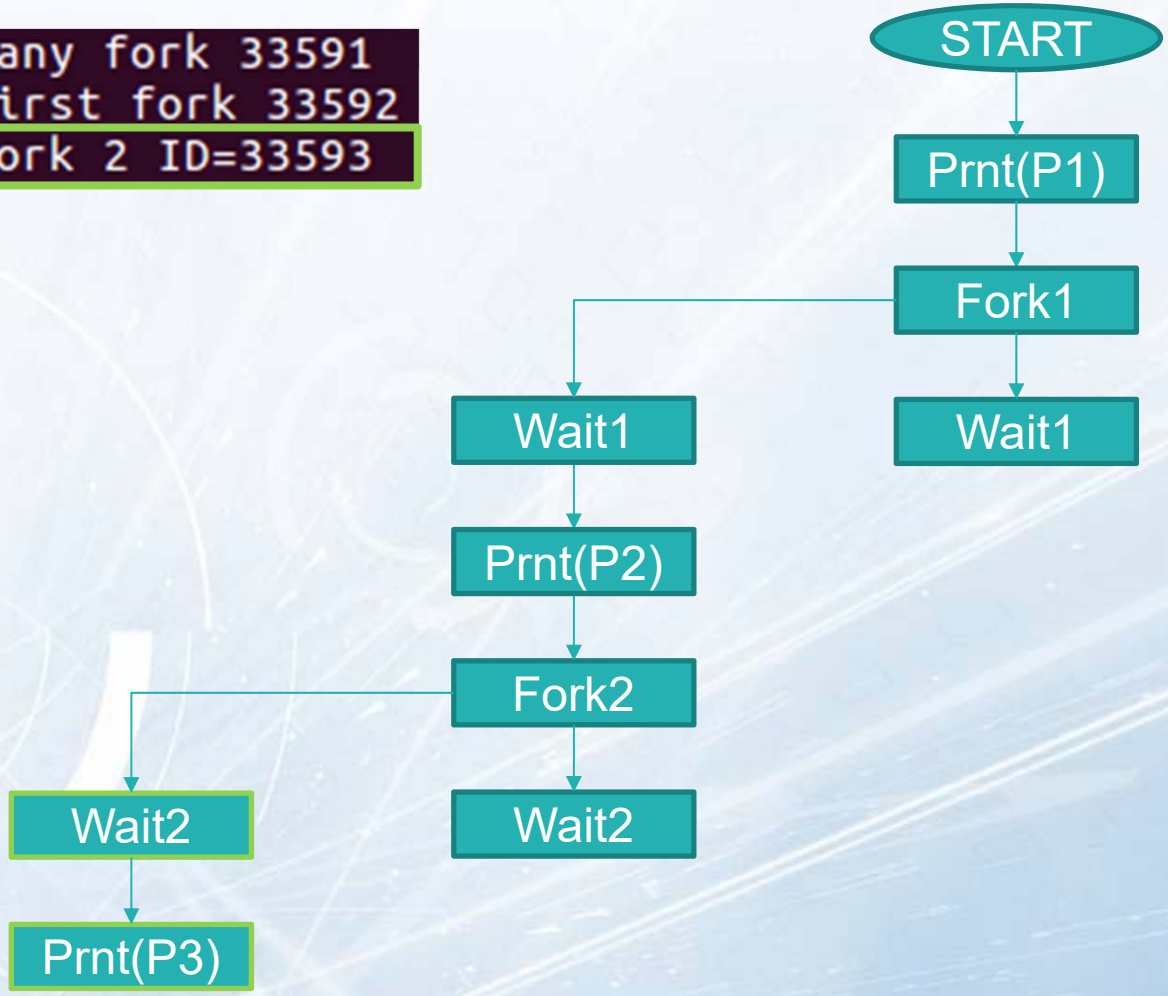



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

3

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
```

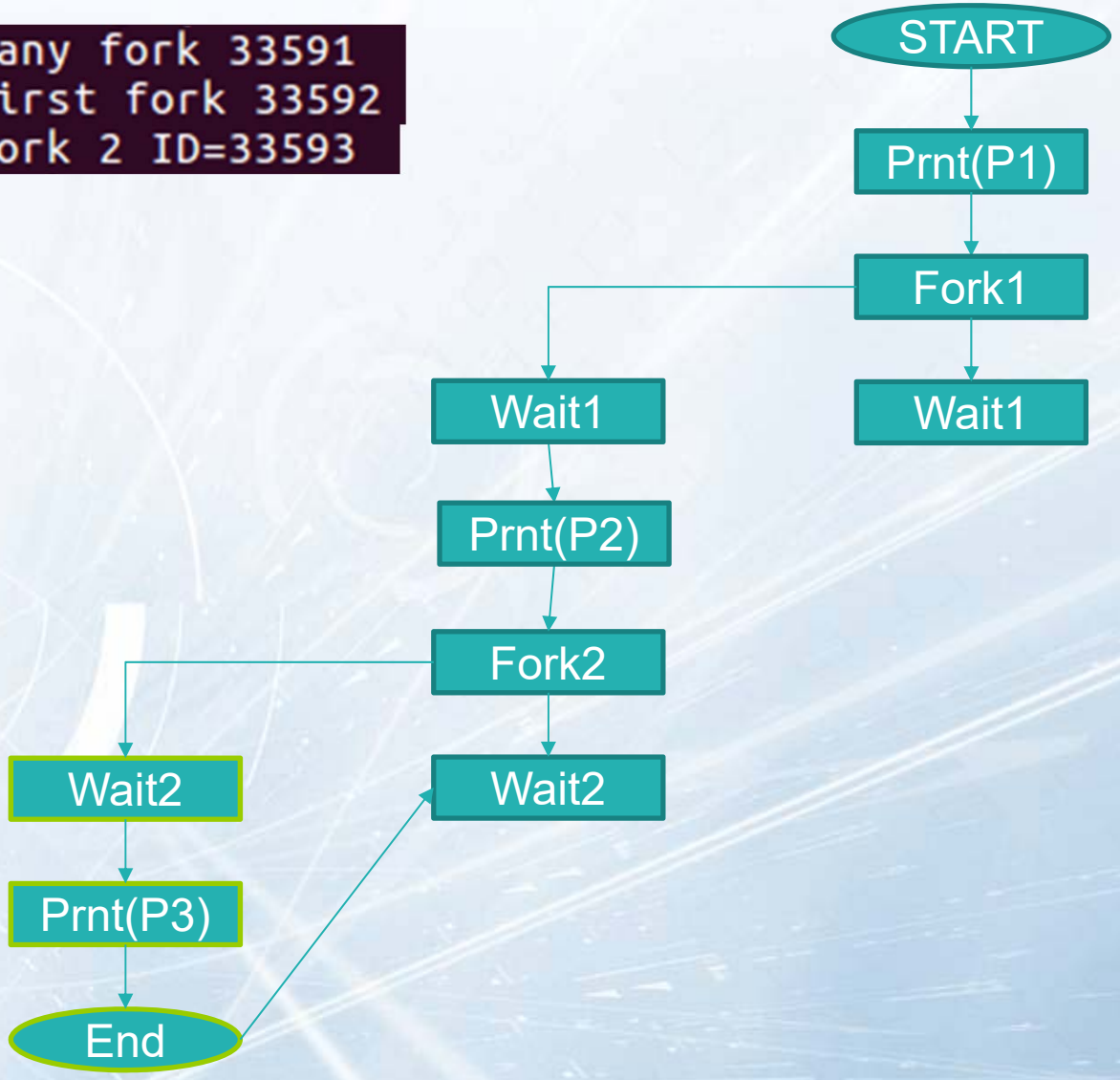



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

3

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
```

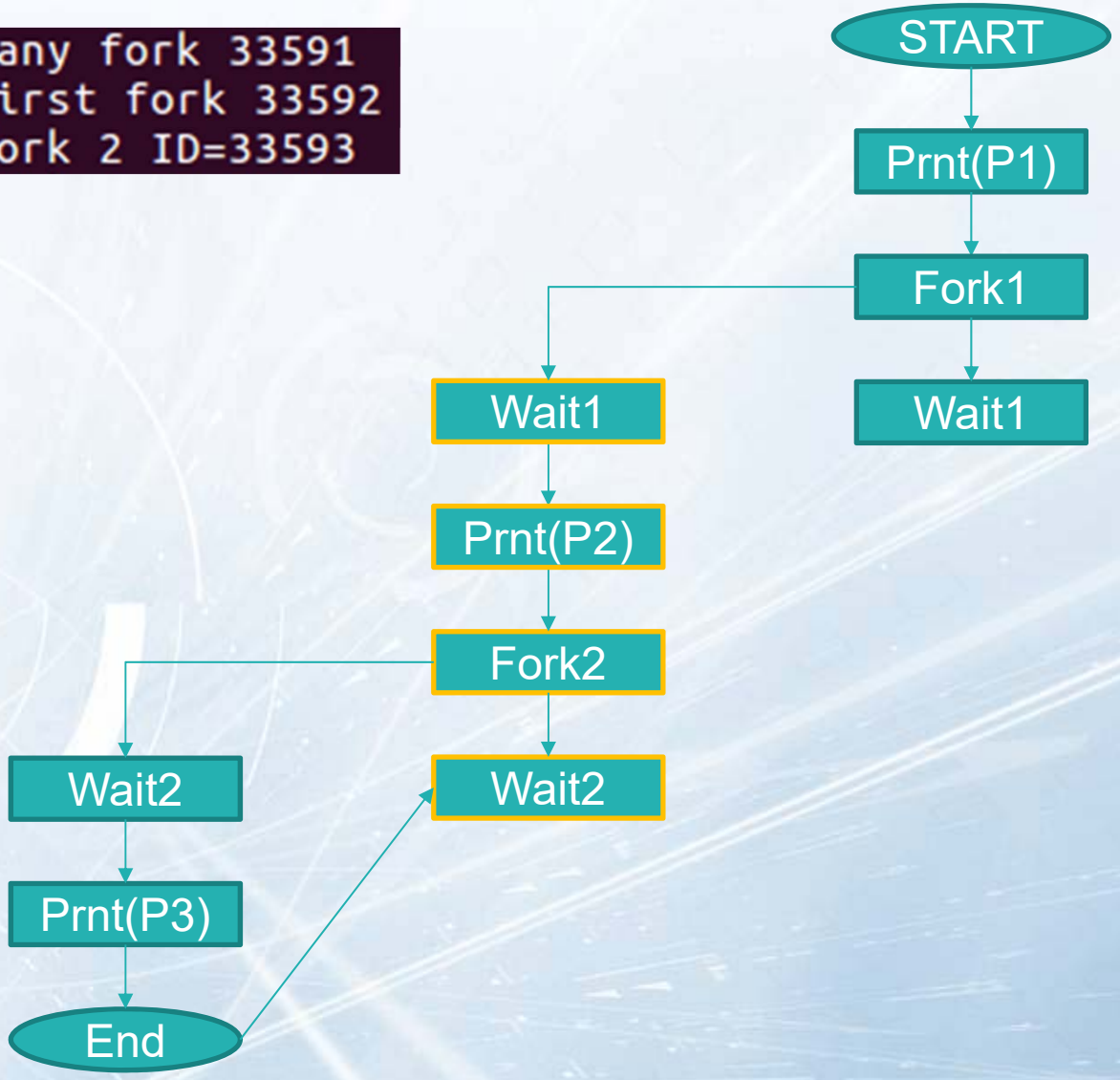



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

2

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
```

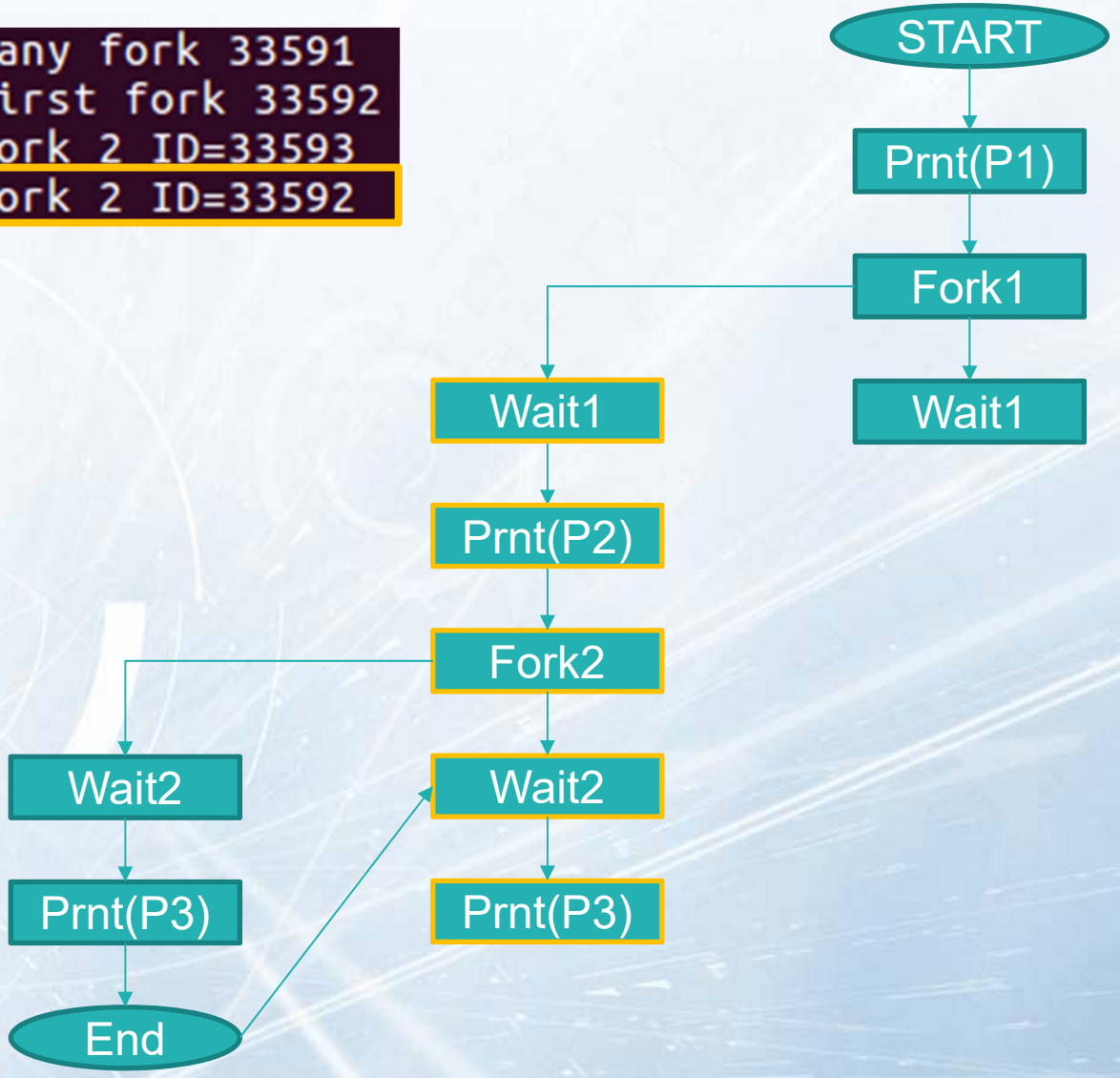



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

2

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
```

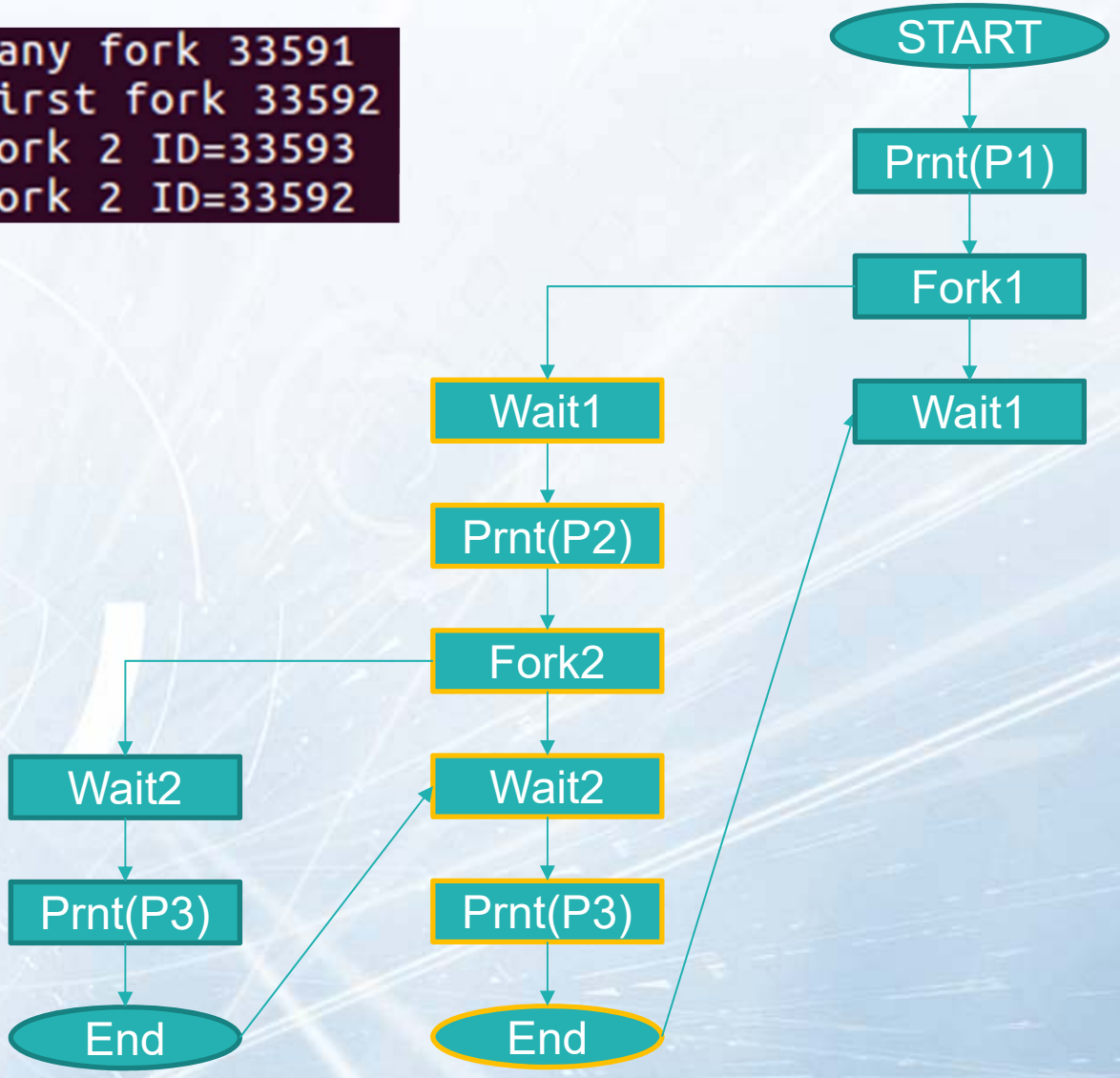



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

2

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
```

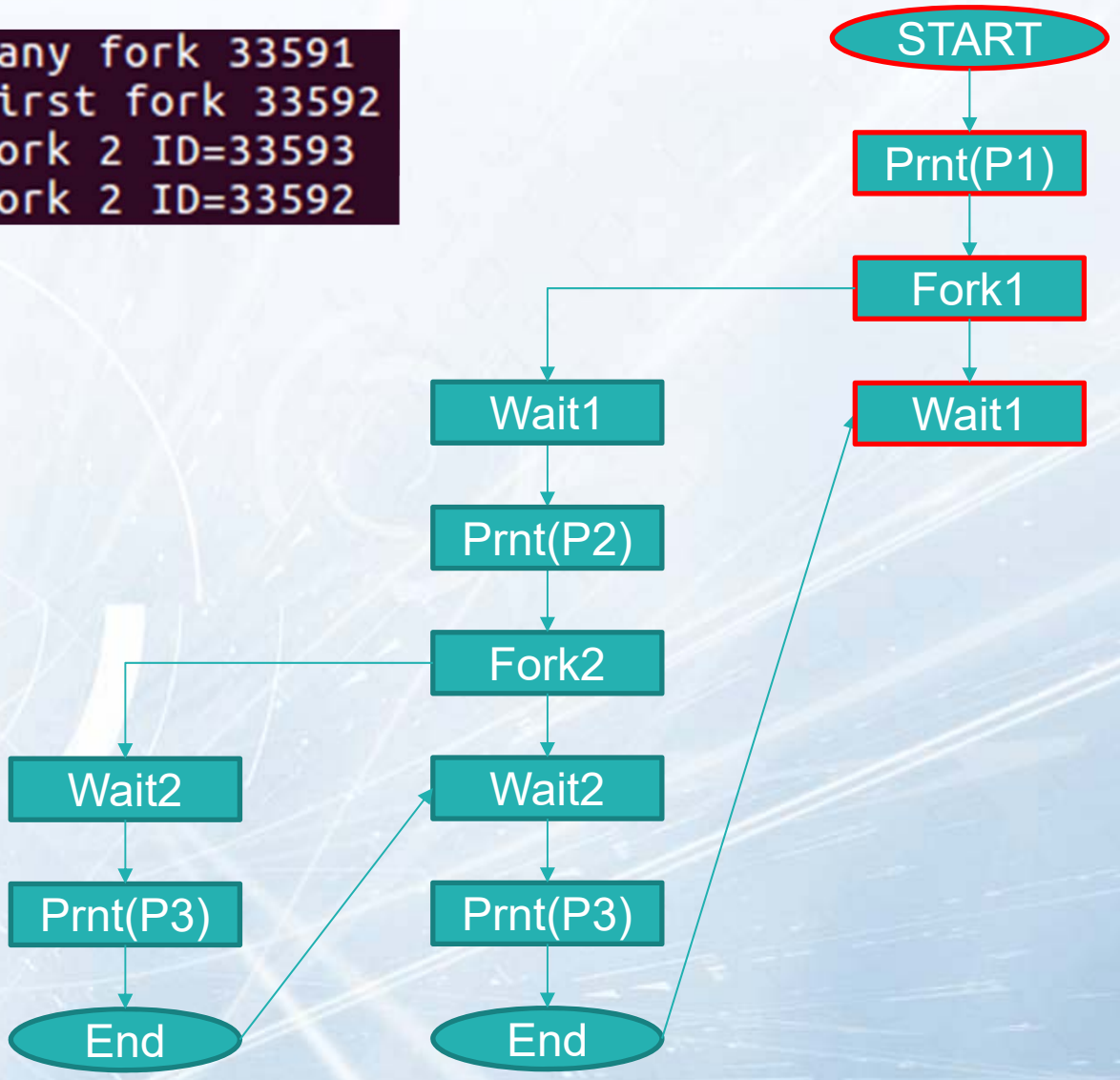



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
```

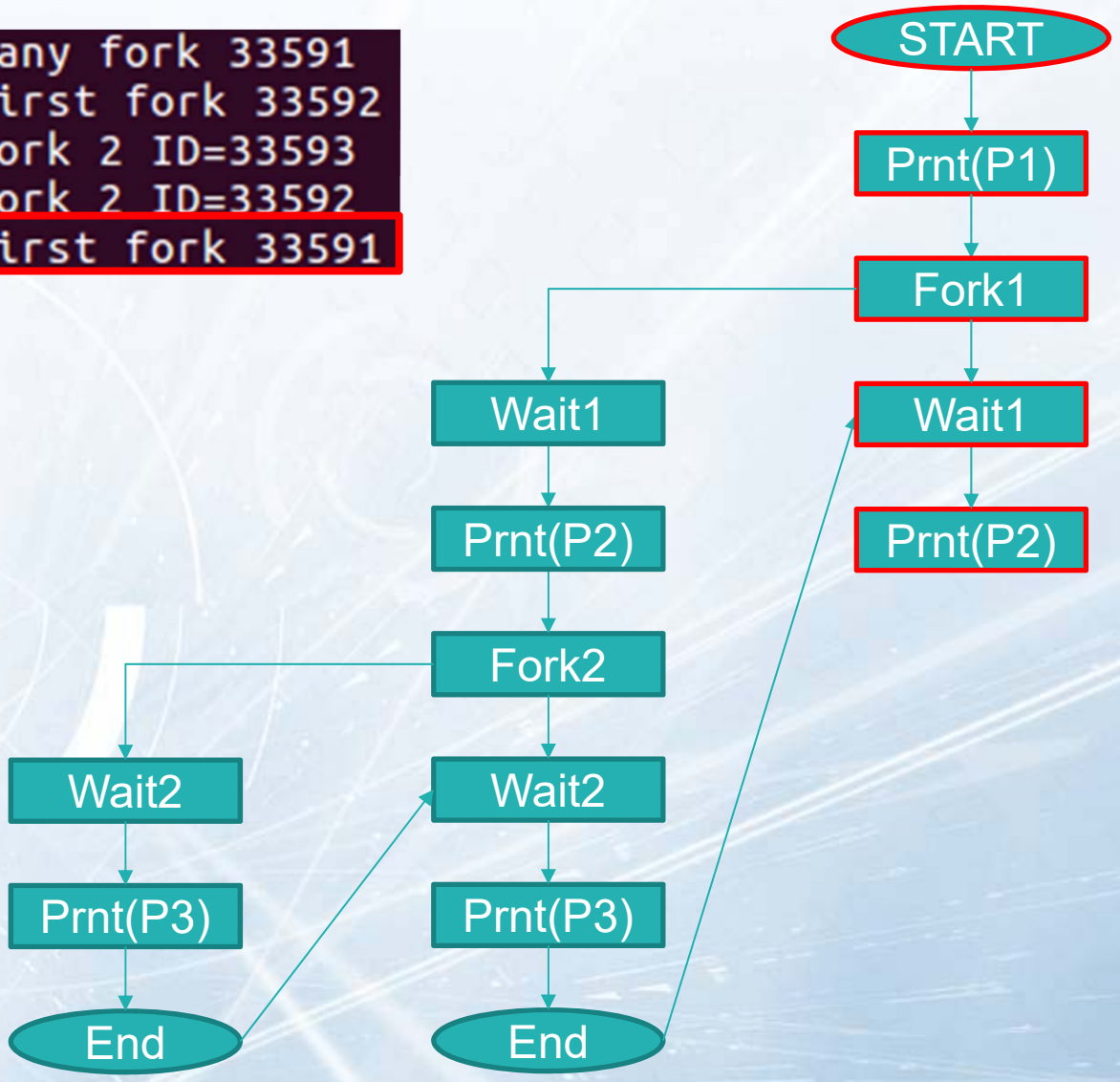



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
```

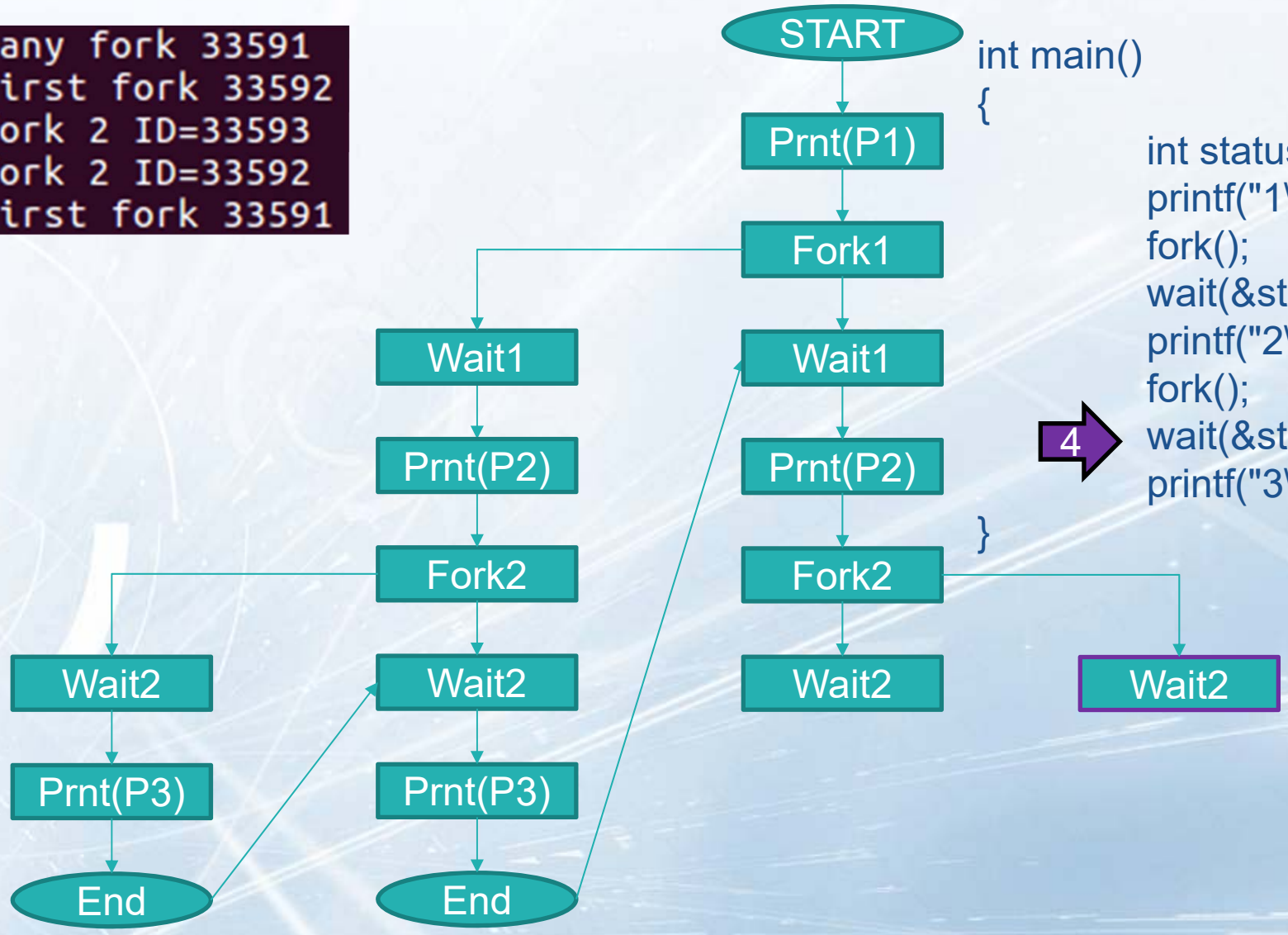



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
```

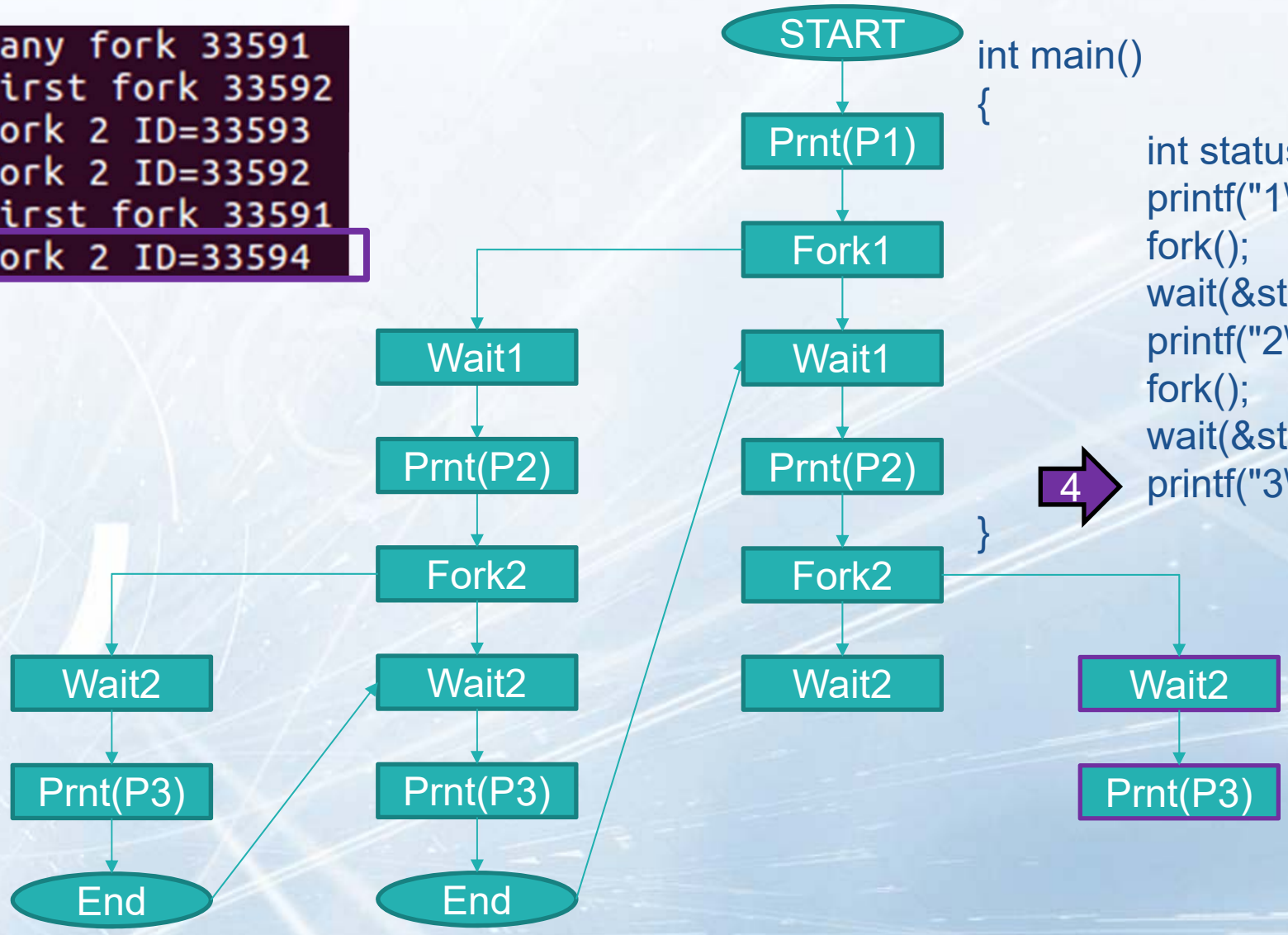



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

4

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
After fork 2 ID=33594
```

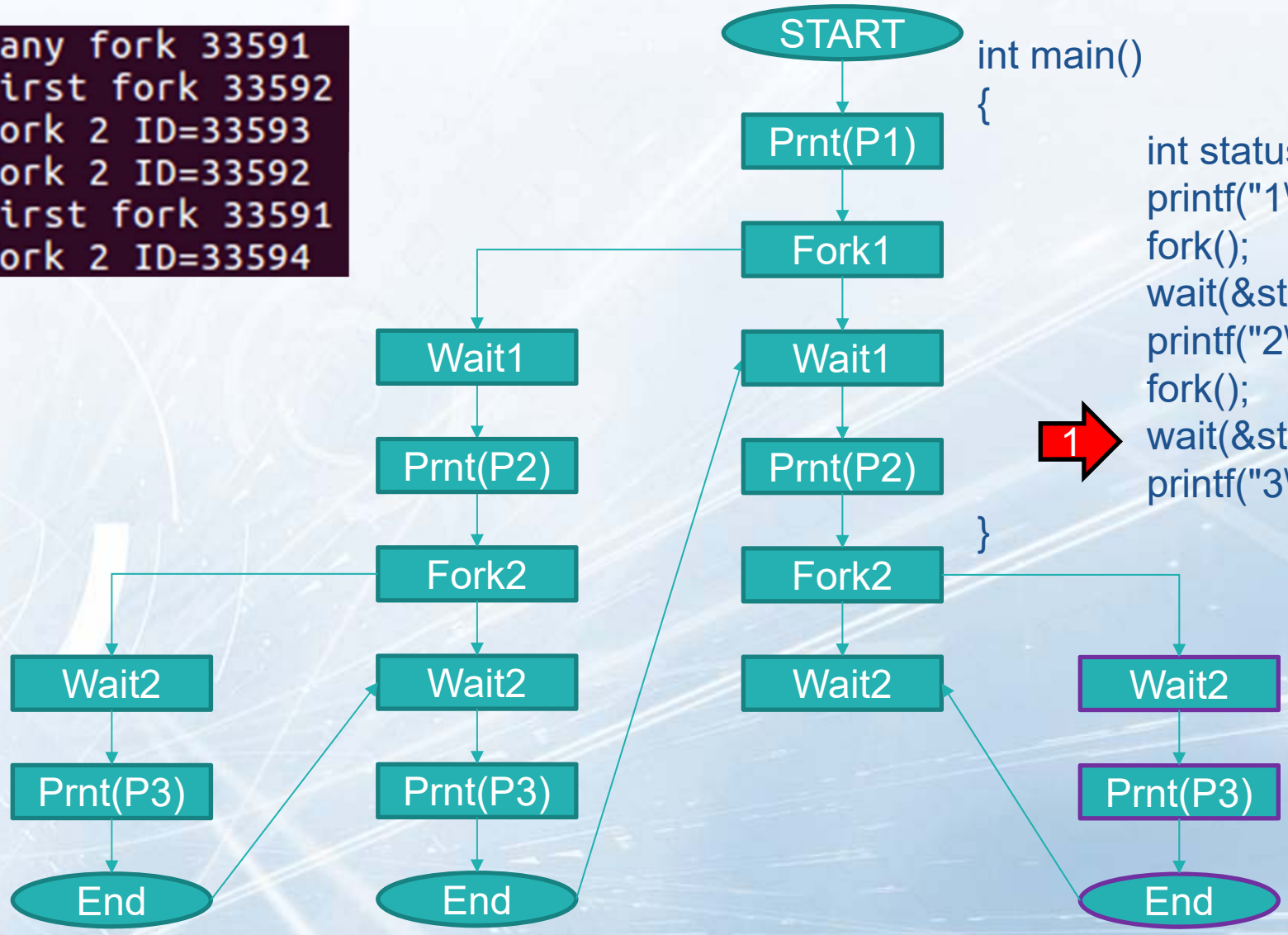



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

4

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
After fork 2 ID=33594
```

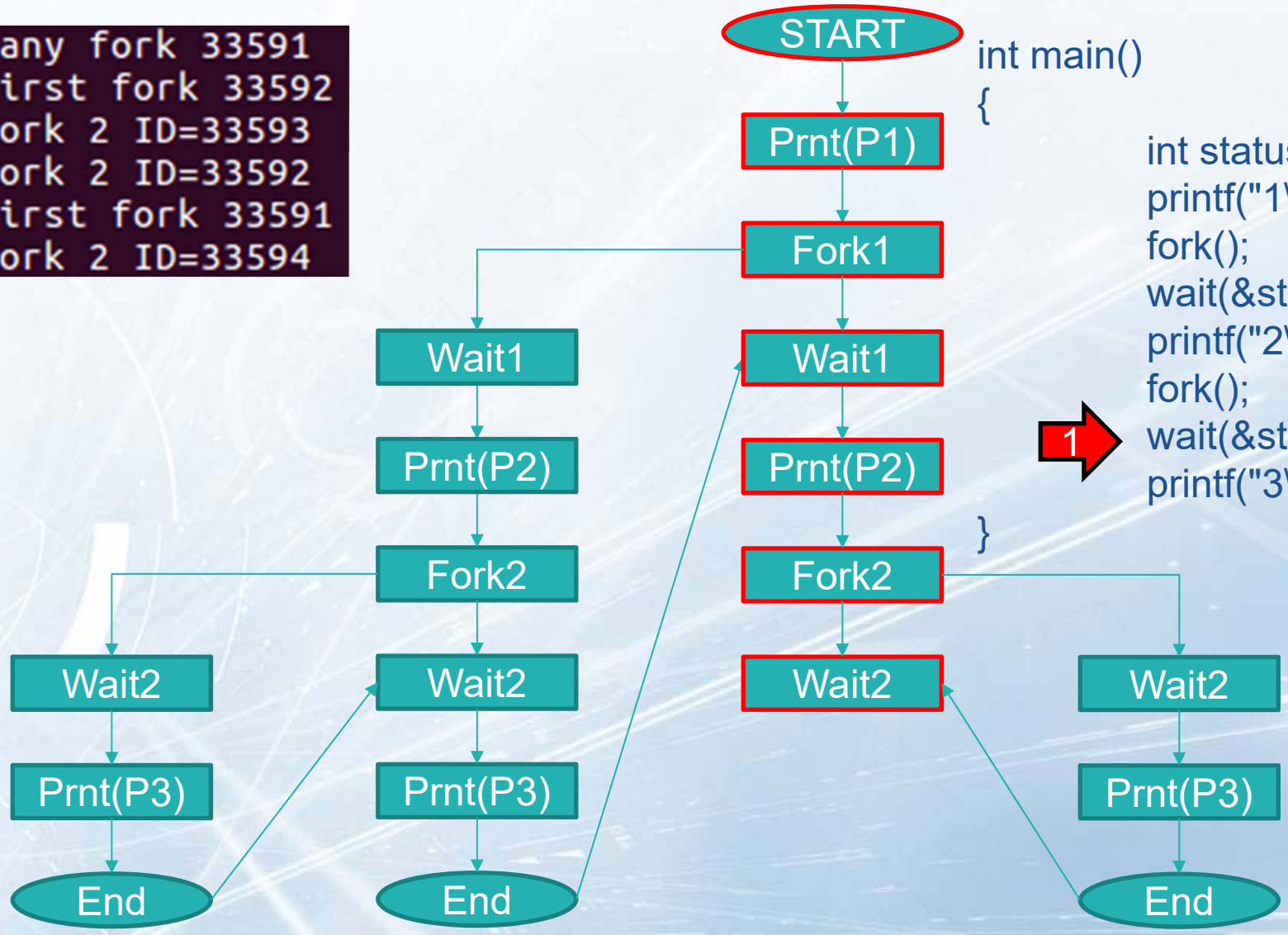



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
After fork 2 ID=33594
```

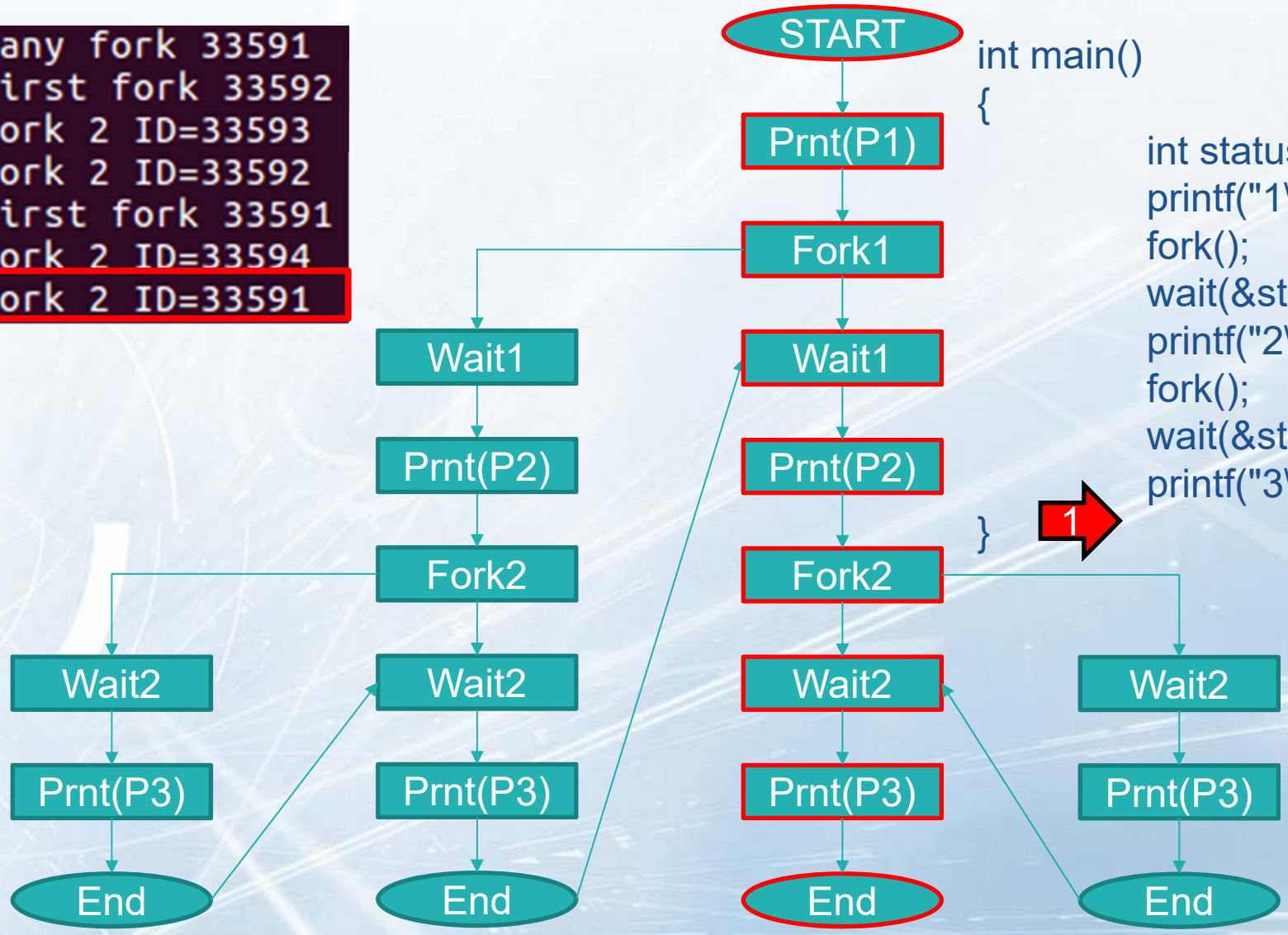



```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

1

שאלה 3

```
before any fork 33591
After first fork 33592
After fork 2 ID=33593
After fork 2 ID=33592
After first fork 33591
After fork 2 ID=33594
After fork 2 ID=33591
```




```
int main()
{
    int status;
    printf("1\n");
    fork();
    wait(&status);
    printf("2\n");
    fork();
    wait(&status);
    printf("3\n");
}
```

שאלה 3

- ומה יקרה עבור הקוד הבא?

```
fork();
fork();
wait(NULL);
```

פתרון- שאלה 3 סעיף אחרון

```
fork();
fork();
wait(NULL);
```

יפתחו 4 תהליכים
כפי הנראה תהליך ילד יסתיים אחרי האבא

```
rmi@ubuntu:~/OS/T13$ ./a.out
1 pid = 21503
2 pid = 21503
3 pid = 21503
3 pid = 21505
4 pid = 21505
4 pid = 21503
rmi@ubuntu:~/OS/T13$ 2 pid = 21504
3 pid = 21504
3 pid = 21506
4 pid = 21506
4 pid = 21504

rmi@ubuntu:~/OS/T13$
```

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/wait.h>


int main()
{
        int status;
        printf("1 pid = %d\n", getpid());
        fork();
        printf("2 pid = %d\n", getpid());
        fork();
        printf("3 pid = %d\n", getpid());
        wait(NULL);
        printf("4 pid = %d\n", getpid());
}
```

שאלה 4

❖מה תהיה התוצאה של הרצת קטע הקוד הבא?

```
int num = 0;
void signal_hand (int sig) {
        signal(SIGUSR1, signal_hand);
        num++;
        printf("num = %d\n", num);
}
void main() {
    i=1;
    while(i<=2){
            if(fork()<0)
                    printf("Error");
            pid_t pid = getpid();
            signal(SIGUSR1, signal_hand);
            kill(pid, SIGUSR1);
            kill(pid, SIGUSR1);
            i++;
    }
}
```

תשובה

במקרה שאין שגיאות:

תהליך ראשי מדפיס:

```
num=1
num=2
num=3
num=4
```

תהליך 1 אשר נוצר על ידי תהליך ראשי מדפיס:

```
num=1
num=2
num=3
num=4
```

תהליך 2 אשר נוצר על ידי תהליך ראשי מדפיס:

```
num=3
num=4
```

תהליך נוסף, אשר נוצר על ידי תהליך 1 מדפיס:

```
num=3
num=4
```