# HW4 xv6 Scheduler

**ID: 208630103, 205493448**

תחילה הוספנו SYSTEMCALL לפי המוסבר בתרגול:

1. הקצנו לכל SYSCALL מספר בתוך -syscall.h
2. הגדרנו את המקום במערך הפונקציות לכל אחת מן הSYSCALLS:
3. הוספנו TRAP חדש לכל אחד מSYSCALL בusys.S
4. הגדרנו את המבנה החדש לפי הנדרש – processInfo
5. הגדרנו את האינטרפייס לuserspace

מימשנו שני :system calls

- getprio המחזיר את ה priority העדכני של התהליך
- setprio המשנה את ה priority של התהליך.

אנו מנהלים את הpriority scheduler בעזרת שני שדות בסטראקט של proc

- Timer_bank – מספר הקובע כמה עוד timer ticks נותר לתהליך לרוץ, מבלי לוותר על הCPU, מספר זה נקבע על ידי <Priority> בכל פעם שהתהליך נבחר על ידי ה scheduler.

```
p->timer_bank = 1 << p->priority;
```

- Priority – מספר בין 0 ל 7, אשר בעזרתנו נקבע את Timer_bank, מספר זה מאותחל כ0 כאשר עושים ()fork, וניתן לקביעה על ידי setprio

```
p->priority = 0; // start priority is 0 by default
```

**אפליקציית ה userspace:**

מימוש ה CPU_BOUND_SLEEP:

```
void sleep_bound_cpu(void)
{
    volatile int x = 0, i;

    for (i = 0; i < SLEEP_COUNTER; i++) {
        x = x + 3.14 * 89.64;
        x /= 6.325;
        x += 123;
        x *= 1.1112;
        x = 10;
    }

}
```

מימוש פונק' העבודה שכל תהליך מבצע:

```
void processWork(int priority)
{
    setprio(priority);

    sleep_bound_cpu();
    printf(1, "Process: %d done, With priority: %d\n", getpid(), getprio());
    exit(); // terminate process

}
```

בפונק' ה()main, מימשנו תהליך אשר מבצע fork 5 פעמים, וכל תהליך בן מריץ את processWork עם priority שונה.

בחרנו את ה priority כך שהfork שמתבצע ראשון הוא בעל הpriority הנמוך ביותר.