



**Department of
Computer Engineering**

Homework 2

Fall 2023

Dr. Javadi

Deadline: 1402/8/26

۱- فرض کنید برنامه ای داریم که با چند نخ در حال اجرا می باشد، با توجه به این موضوع درستی و نادرستی عبارات زیر را با ذکر دلیل مشخص کنید و توضیح دهید.

الف) اگر یک نخ آرگومان های خاصی را به یک تابع در برنامه ارسال کند، این آرگومان ها برای نخ های دیگر قابل مشاهده هست.

ب) اگر یک نخ با استفاده از دستور malloc حافظه اضافی به خود تخصیص دهد می تواند باعث به وجود آمدن خطای out of memory برای نخ دیگری در برنامه شود.

پ) رشته های سطح کاربر توسط کتابخانه سطح کاربر برنامه ریزی می شوند و بدون اینکه هسته از عملیات آنها مطلع باشد کار می کنند.

ج) زمان context switch در رابطه با نخ های سطح هسته کمتر طول می کشد .

د) نخ های سطح کاربر نمی توانند به صورت موازی واقعی (true parallelism) در سیستم های چند هسته ای اجرا شوند زیرا توسط یک رشته در سطح هسته مدیریت می شوند.

۲- خروجی قطعه کد زیر را پیش بینی کنید و پیش بینی خود را توضیح دهید. سپس به سوالات زیر پاسخ دهید.

```
#define NUM_THREADS 3
int shared_value = 10;

void* increaseValue (void* threadID) {
    long tid = (long)threadID;
    printf("Thread %ld:shared value before increament:%d\n", tid, shared_value);
    shared_value += 5;
    printf("Thread %ld:shared value after increament:%d\n", tid, shared_value);
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    long t;
    for (t = 0; t < NUM_THREADS; t++) {
        printf("creating thread %ld\n", t);
        pthread_create(&threads[t], NULL, increaseValue, (void*)t);
    }
    for (t = 0; t < NUM_THREADS; t++) {
        pthread_join(threads[t], NULL);
    }
    printf("All threads have completed. Final shared value: %d\n", shared_value);
    pthread_exit(NULL);
    return 0;
}
```

الف) در صورتی که دو تابع `pthread_create` و `pthread_join` در یک حلقه صدا زده شوند. خروجی برنامه چه خواهد بود و علت آن را توضیح دهید.

ب) تفاوت بین استفاده از رشته ها و استفاده از `fork` برای رسیدن به موازی سازی را توضیح دهید.

ج) آیا چند نخ می توانند تابع `pthread_join` را بر روی یک نخ هدف فراخوانی کنند یا محدودیتی وجود دارد که تنها یک نخ انتظارها را برای هر نخ هدف می تواند داشته باشد؟

۳- با توجه به برنامه زیر، به موارد خواسته شده پاسخ دهید. (فرض کنید ترد ها در ابتدا به ترتیب شروع می شوند.)

```
P1:
fork()
func1:
    print("x")
func2:
    fork()
t1 = pthread(func1)
t2 = pthread(func2)
t3 = pthread(func1)
join t1, t2, t3
wait()
```

الف) تعداد x های چاپ شده بعد از اتمام برنامه

ب) تعداد پردازش ها بعد از اتمام برنامه

ج) تعداد ترد ها بعد از اتمام برنامه

د) رسم درخت پردازش ها و مشخص کردن تعداد ترد های هرکدام

ه) خلاصه از فرآیند را شرح دهید

۴- با فرض موفقیت آمیز بودن اجرای دستورات `fork` و `execv`، خروجی قطعه کد زیر را به صورت دقیق و با ذکر دلیل

بیان کنید.

```
int main(){
    pid_t pid;
    pid = fork()
    if(pid == 0){
        printf("process 1\n");
        char* args[] = {"ls", "-1", NULL};
        execv("/bin/ls", args);
        printf("process 1 finished\n");
    }
    else if(pid > 0){
        printf("process 2\n");
        wait(NULL);
        printf("process 1 terminated\n");
    }
    return 0;
}
```

۵- با فرض آنکه پردازش‌های producer و consumer به نحو زیر پیاده سازی شده اند، اگر سایز buffer برابر با 5 باشد و متغیرهای in و out در ابتدا برابر با 0 باشند، در هر مورد خروجی را با ذکر دلیل مشخص کنید.

Producer:

```
int next_produced = 0;
while(next_produced < 10) {
    buffer[in] = ++next_produced;
    in = (in + 1) % BUFFER_SIZE;
}
```

Consumer:

```
int next_consumed, sum;
while(next_consumed < 10) {
    if(in == out){
        continue;
    }
    next_consumed = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    sum += next_consumed;
}
Printf("%d", sum);
```

الف) به ازای هر یکبار اجرا بدنه حلقه producer یک بار بدنه حلقه consumer اجرا شود.

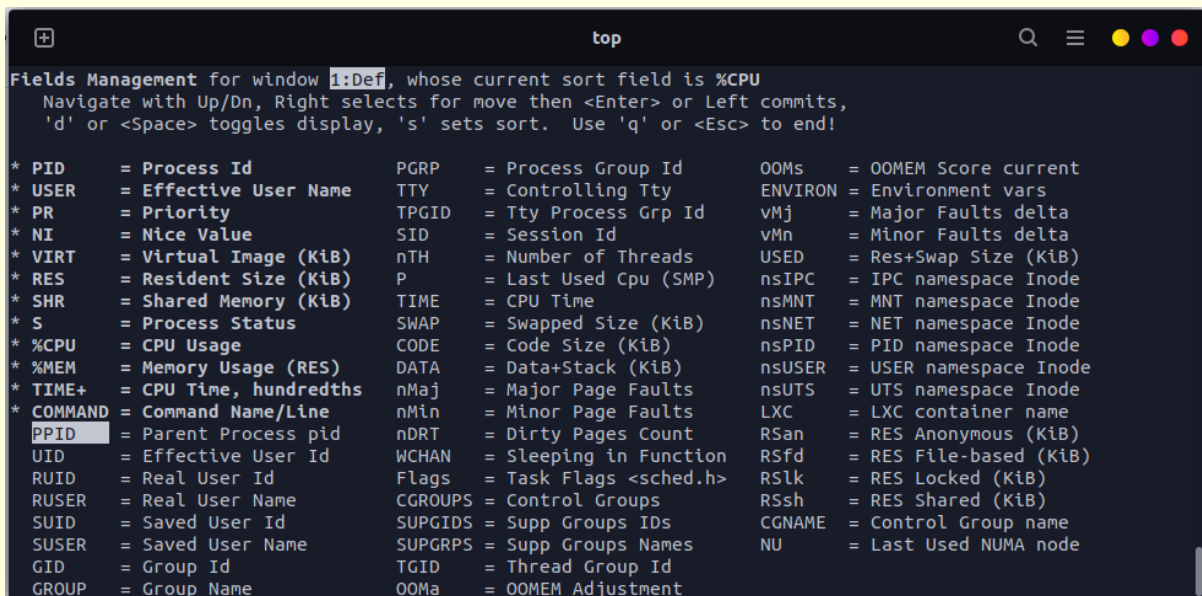
ب) به ازای هر دوبار اجرا بدنه حلقه producer یک بار بدنه حلقه consumer اجرا شود.

پ) به ازای هر سه بار اجرا بدنه حلقه producer یک بار بدنه حلقه consumer اجرا شود.

۶- تا به اینجا درس با موازی سازی فرایندها آشنا شده اید. یکی از مفاهیم در این خصوص multi-core است. برای مشاهده مشخصات پردازنده میتوان از دستور **lscpu** استفاده کرد این دستور، اطلاعات پردازنده را از فایل `proc/cpuinfo` می خواند و در ساختاری مناسب به نمایش می گذارد . خروجی دستور **lscpu** را در گزارش خود قرار دهید.

دستور **top** یکی از دستورات کاربردی در مدیریت پردازنده هاست. این دستور اطلاعات مربوط به thread های پردازنده، شماره هسته پردازنده، درصد استفاده از هر هسته پردازنده، میزان استفاده هر پردازنده از کل cpu و ... را به ما نمایش می دهد.

این دستور به صورت پیش فرض همه اطلاعات درمورد پردازنده ها را به ما نمایش نمی دهد اما میتوانیم به صورت دستی خودمان یک سری از اطلاعاتی که نیاز داریم را به ستون های اطلاعاتش اضافه و کم کنیم. برای اینکار پس از اجرای دستور **top** دکمه f را فشار دهید. صفحه ای مشابه تصویر زیر برایتان نمایش داده میشود . با کلید های بالا و پایین کیبورد میتوان در این گزینه ها جا به جا شد و با کلید space میتوان گزینه های مختلفی را انتخاب کرد.



```

+ top
Fields Management for window 1:Def, whose current sort field is %CPU
Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
'd' or <Space> toggles display, 's' sets sort. Use 'q' or <Esc> to end!

* PID      = Process Id          PGRP      = Process Group Id      OOMs      = OOMEM Score current
* USER     = Effective User Name TTY       = Controlling Tty          ENVIRON   = Environment vars
* PR       = Priority           TPGID     = Tty Process Grp Id   VMj       = Major Faults delta
* NI       = Nice Value        SID       = Session Id           VMn       = Minor Faults delta
* VIRT     = Virtual Image (KiB) nTH       = Number of Threads       USED      = Res+Swap Size (KiB)
* RES      = Resident Size (KiB) P         = Last Used Cpu (SMP)      nsIPC     = IPC namespace Inode
* SHR      = Shared Memory (KiB) TIME      = CPU Time                nsMNT     = MNT namespace Inode
* S        = Process Status    SWAP      = Swapped Size (KiB) nsNET     = NET namespace Inode
* %CPU     = CPU Usage         CODE      = Code Size (KiB)   nsPID     = PID namespace Inode
* %MEM     = Memory Usage (RES) DATA     = Data+Stack (KiB)       nsUSER    = USER namespace Inode
* TIME+    = CPU Time, hundredths nMaj      = Major Page Faults      nsUTS     = UTS namespace Inode
* COMMAND  = Command Name/Line nMin      = Minor Page Faults      LXC       = LXC container name
* PPID     = Parent Process pid nDRT      = Dirty Pages Count      RSAn      = RES Anonymous (KiB)
* UID      = Effective User Id  WCHAN     = Sleeping in Function  RSfd      = RES File-based (KiB)
* RUID     = Real User Id      Flags     = Task Flags <sched.h>  RSlk      = RES Locked (KiB)
* RUSER    = Real User Name    CGROUPS   = Control Groups       RSsh      = RES Shared (KiB)
* SUID     = Saved User Id    SUPGIDS   = Supp Groups IDs      CGNAME    = Control Group name
* SUSER    = Saved User Name  SUPGRPS   = Supp Groups Names  NU        = Last Used NUMA node
* GID      = Group Id         TGID      = Thread Group Id
* GROUP    = Group Name       OOMa      = OOMEM Adjustment
  
```

nTH تعداد thread های یک پردازنده و P شماره آخرین هسته پردازنده ای که برای پردازنده استفاده شده است را نمایش می دهد. این دو گزینه را انتخاب کنید و خروجی آن را در گزارشتان بیاورید. همچنین با فشار دادن کلید 1 در پردازنده **top** می توان میزان استفاده از هر کدام از core های پردازنده را مشاهده کرد.

گاهی اوقات می خواهیم که یک پردازنده، برای مدیریت بهتر، بر روی یک core از پردازنده اجرا شود. با دستور **taskset** میتوانیم مشخص کنیم که پردازنده روی کدام یک از core های پردازنده اجرا شود.

دستور زیر را برای یک core دلخواه و یک دستور دلخواه اجرا کرده و نتیجه آن را در خروجی دستور **top**، در گزارش بیاورید. (امتیاز ۵)

```
~ ▶ taskset --cpu-list CPU_NUMBER COMMAND
```

نکات مهم:

*مهلت ارسال تمرین ساعت 23:59 روز 1402/8/26 می باشد، با توجه به مهلت تجمیعی هفت روز تاخیر مجاز برای تمارین، امکان تمدید تمرین وجود ندارد، بنابراین توصیه می شود ارسال پاسخ های خود را به ساعات پایانی موکول نکنید.

*در صورت کشف هر گونه تقلب بار اول برای هر دو طرف نمره صفر لحاظ می شود و از دفعات بعد مشمول جریمه نیز می گردند.

*پاسخ های خود را در قالب یک فایل pdf یا zip با فرمت OS_HW2_StudentNumber.pdf یا OS_HW2_StudentNumber.zip ارسال نمایید، مانند:

OS_HW2_9931005.pdf

*در تمام سوالات پیاده سازی توابع fork و ... را مطابق با مطالب آموزش داده شده در کلاس در نظر بگیرید و اجرای آنها را موفقیت آمیز و بدون خطا لحاظ کنید.

*هر گونه سوال خود را می توانید در گروه پرسش و پاسخ درس از ما بپرسید.

Useful links for study:

<https://www.geeksforgeeks.org/thread-functions-in-c-c/>

<https://man7.org/linux/man-pages/man1/lscpu.1.html>

<https://www.geeksforgeeks.org/top-command-in-linux-with-examples/>

<https://www.geeksforgeeks.org/difference-fork-exec/>

<https://linuxhint.com/use-taskset-command/>