

2020 OS Project 1 - Process Scheduling

資工二 b07902078 沈韋辰

1. Kernel Version

Kernel的部分本次選用linux4.10.13，在數次嘗試中發現，太新的版本會有一些不知名的bug

2. Design

每筆資料的處理方式如下:

```
./scheduler  
=> ./FIFO Or ./RR Or ./SJF Or ./PSJF  
=> fork()  
=> run the process  
=> send a signal to parent
```

(1) Initialization

```
USE_CPU(0);  
INIT_DATA();  
qsort(pcb, N, sizeof(PCB), compare);  
signal(SIGFNS, signal_routine);  
int pcbptr = 0;  
enable = TRUE;  
finish = 0;
```

- PCB: store ready, run time, pid, etc.
- USE_CPU(x): set the cpu affinity
- qsort(...): sort the process by ready time
- signal_routine: my signal handler
- enable: a lock to determine whether CPU is available
- finish: counter of the finished task

(2-1) fork()-child

```
// child process
if (PID == 0) {
    USE_CPU(1);
    for (int i = 0; i < timeAmount; i++)
        unit_time();
    if(pcb->runtime == timeAmount)
        kill(ppid, SIGFNS);
    else
        kill(ppid, SIGTERM);
    exit(0);
}
```

- USE_CPU(1): child use another CPU
- unit_time(): do 1000000 times for-loop
- SIGFNS: the process has finished
- SIGTERM: the process hasn't finished

(2-2) fork()-parent

```
// parent process
else if (PID > 0) {
    if(pcb->pid == 0)    pcb->pid = PID;
    return;
}
```

- when we first run this process, store the PID.

(3) signal handler

```
void signal_routine(int signo){
    if(signo == SIGFNS){
        #ifdef SYSCALL_AVAILABLE
            runNow->end_time = syscall(SYS_TIME);
            syscall(SYS_PRINTK, runNow->pid, runNow->start_time, runNow->end_time);
        #else
            runNow->end_time = (Long) time(NULL);
            fprintf(stderr, "[Project1] %d %ld %ld\n", runNow->pid, runNow->start_time, runNow->end_time);
        #endif

        enable = TRUE;
        finish++;
    }

    else if(signo == SIGTERM)
        enable = TRUE;

    return;
}
```

- SYSCALL_AVAILABLE: defined if run in syscall installed environment

(4-1) system call - pjtimer

```
// syscall #548
#include <linux/linkage.h>
#include <linux/kernel.h>
#include <linux/timer.h>

asm linkage long sys_pjtimer(void) {
    const long BASE = 1e9;
    struct timespec t;
    getnstimeofday(&t);
    return t.tv_sec * BASE + t.tv_nsec;
}
```

(4-2) system call - pjprint

```
// syscall #549
#include <linux/linkage.h>
#include <linux/kernel.h>

asm linkage void sys_pjprint(int pid, long start, long end) {
    const long BASE = 1e9;
    if(pid==0)
        printk(KERN_INFO "[RELEASE]");
    else
        printk(KERN_INFO "[Project1] %d %ld.%09ld %ld.%09ld", pid, start / BASE, start % BASE, end / BASE, end % BASE);
}
```

- [Project1] tag - 紀錄process完成
- [RELEASE] tag - 另有特殊用途，將由下則說明

(4-3) system call - append on the end of policy

- 此為本人在作業中發生的一個問題: 同時發出stderr和sys_pjprint時，兩者得到的數據不同，似乎平移了一個進程。
- 經過本人多次實驗後發現，每個policy發出的最後一個printk會被block住，直到下一個case呼叫時，上一個case的最後一個printk message則會被呼叫，並且會讓之後的訊息全部被往後推一個進程，連帶影響後續的結果。

(下圖的上半部stderr, 下半部則是dmesg, 可見兩者的PID有錯位發生)

```
[Project1] 3919 1588197693593168534 1588197693716957645
[Project1] 3920 1588197693788584093 1588197693882697892
[Project1] 3921 1588197693978023650 1588197694071565142
[Project1] 3922 1588197694166963218 1588197694268835378
[Project1] 3923 1588197694360534466 1588197694455939641
[Project1] 3924 1588197694548525404 1588197694642563756
[Project1] 3925 1588197694737516460 1588197694838335837
[Project1] 3926 1588197694931959621 1588197695025765366
[Project1] 3927 1588197695119113391 1588197695216292364
[Project1] 3928 1588197695310592299 1588197695404519951
[Project1] 3928 1588197695310592299 1588197695404519951
root@watson-VirtualBox:~/Downloads/[BLOCKED]-OSproject1# dmesg
[ 1753.120517] [Project1] 3680 1588196992.301116573 1588196992.406374771
[ 2456.108307] [Project1] 3919 1588197693.593168534 1588197693.716957645
[ 2456.108319] [Project1] 3920 1588197693.788584093 1588197693.882697892
[ 2456.108349] [Project1] 3921 1588197693.978023650 1588197694.071565142
[ 2456.108356] [Project1] 3922 1588197694.166963218 1588197694.268835378
[ 2456.108361] [Project1] 3923 1588197694.360534466 1588197694.455939641
[ 2456.108366] [Project1] 3924 1588197694.548525404 1588197694.642563756
[ 2456.108371] [Project1] 3925 1588197694.737516460 1588197694.838335837
[ 2456.108377] [Project1] 3926 1588197694.931959621 1588197695.025765366
[ 2456.108385] [Project1] 3927 1588197695.119113391 1588197695.216292364
```

- 然而，printk為Non-blocking的system call，因此其無法正常運作的原因仍不明(maybe: kernel version, race condition, etc)
- 但實做上，我們可以在pjprint內加一個用來release那個無法成功的printk，類似用自身卡位使得每次做出N+1個printk，這樣最後輸出的release反而成為下一個的第一個
- 我們會將下面這個system call 家在所有policy的最後一行

```
syscall(SYS_PRINTK, 0, 0, 0);
```

而每個Policy大致相同，只有以下部分有所區別:

a. First in, first served (FCFS)

```
for(int cur=0 ; finish!=N ; cur++){
    if(cur >= pcb[pcbptr].readytime && enable==TRUE ){
        runNow = &pcb[pcbptr];
        enable = FALSE;
        pcbptr++;
        create_process( runNow, runNow->runtime);
    }
    unit_time();
}
```

- 目標: 先進來的先跑
- 直接把pcb當作ready queue, 當process離開queue時，enable就設成FALSE(鎖門)
- enable設成TRUE(開門)的方式是使用signal handler來開鎖

b. Round Robin (RR)

```

for(int cur=0 ; finish != N ; cur++){

    if( enable == TRUE){
        invalid = 0;
        while (invalid != N) {
            if(pcb[pcbptr].runtime == 0 || pcb[pcbptr].readytime > cur ){
                pcbptr = (pcbptr + 1) % N;
                invalid++;
            }
            else
                break;
        }
        if(invalid == N)
            continue;

        runNow = &pcb[pcbptr];
        enable = FALSE;
        int timeAmount = (runNow->runtime > time_slice) ? time_slice : runNow->runtime;
        create_process( runNow, timeAmount);
        pcb[pcbptr].runtime -= timeAmount;
        pcbptr = (pcbptr + 1) % N;
    }
    unit_time();
}

```

- 上半部(continue之前): polling一輪去找下個ready的process。為了以防陷入無限迴圈，用invalid去紀錄有幾個現在不能跑(跑完or還沒ready)
- 下半部(continue之後) 每次能跑的上限時間是給定的time slice，跑完後計算尚需的時間。
- 整體: SIGFNS和SIGTERM都可以開鎖，但SIGFNS會去increase finish的數量。

c. Shortest Job First (SJF)

```

for(int cur=0 ; finish!=N ; cur++){

    while(pcbptr < N && pcb[pcbptr].readytime == cur){
        Insert(Heap,&size,&pcb[pcbptr]);
        pcbptr++;
    }

    if( size != 0 && enable == TRUE){
        runNow = Extract(Heap, &size);
        enable = FALSE;
        create_process(runNow, runNow->runtime);
    }

    unit_time();
}

```

- 用一個heap來maintain processes，heap的最上面都是run time最小的(runtime一樣的話就看誰先來)
- 只要ready time到了可以就insert到heap裡，每次開鎖時就extract一個去跑。

d. Preemptive Shortest Job First (PSJF)


```

for(int cur=0 ; finish!=N ; cur++){

    while(pcbptr < N && pcb[pcbptr].readytime == cur){
        Insert(Heap, &size, &pcb[pcbptr]);
        pcbptr++;
    }

    if( size != 0 && enable == TRUE){
        runNow = Extract(Heap, &size);
        if(yet != NULL){
            Insert(Heap, &size, yet);
            yet = NULL;
        }
        enable = FALSE;
        int timeAmount = (runNow->runtime > time_slice) ? time_slice : runNow->runtime;
        create_process(runNow, timeAmount);
        runNow->runtime -= timeAmount;

        if(runNow->runtime != 0){
            if(size == 0)
                Insert(Heap, &size, runNow);
            else
                yet = runNow;
        }
    }

    unit_time();
}

```

- 上半部 (while-loop): 跟SJF一樣，用一個heap去排ready的processes.
- 下半部 (大if裡面): 當enable且heap不為空時，就從頂端拿一個來跑一段時間(time slice 與 run time較小的那個):

1. 跑完了 => do nothing
2. 還沒跑完，heap有人 => 把自己放到yet，下一個extract後再把自己insert進去
3. 還沒跑完，heap沒人 => 沒人也要跑那就可以繼續跑

3. Comparison

每一筆測資都有.txt, stdout.txt 和 dmesg.txt (如下圖由左到右為TIME_MEASUREMENT的這三筆資料)

FIFO	10	P0 0 500	P0 2919	[3950.678015]	[Project1]	2919	1588215971.729562779	1588215971.904987448
P1 1000 500	P1 2920	[3950.989864]	[Project1]	2920	1588215972.039759273	1588215972.216842060		
P2 2000 500	P2 2921	[3951.272003]	[Project1]	2921	1588215972.360695866	1588215972.498987976		
P3 3000 500	P3 2922	[3951.548143]	[Project1]	2922	1588215972.636846596	1588215972.775134620		
P4 4000 500	P4 2923	[3951.828633]	[Project1]	2923	1588215972.917288568	1588215973.055631468		
P5 5000 500	P5 2924	[3952.136663]	[Project1]	2924	1588215973.190480992	1588215973.363668699		
P6 6000 500	P6 2925	[3952.450114]	[Project1]	2925	1588215973.538353223	1588215973.677125753		
P7 7000 500	P7 2926	[3952.716743]	[Project1]	2926	1588215973.811127259	1588215973.943761682		
P8 8000 500	P8 2927	[3952.987300]	[Project1]	2927	1588215974.081406289	1588215974.214324327		
P9 9000 500	P9 2928	[3953.252338]	[Project1]	2928	1588215974.346620678	1588215974.479368196		

由於為了節省空間，下方的每個比較將使用以下這個我自己寫的table以方便比較，如：

Name	PID	Ready	Run	Start Time	End Time	Time interval
P0	2919	0	500	1588215971.7295628	1588215971.9049873	0.17542457580566406
P1	2920	1000	500	1588215972.0397592	1588215972.2168422	0.17708301544189453
P2	2921	2000	500	1588215972.3606958	1588215972.498988	0.1382920742034912
P3	2922	3000	500	1588215972.6368465	1588215972.7751346	0.13828802108764648
P4	2923	4000	500	1588215972.9172885	1588215973.0556314	0.13834285736083984
P5	2924	5000	500	1588215973.190481	1588215973.3636687	0.1731877326965332
P6	2925	6000	500	1588215973.5383532	1588215973.6771257	0.13877248764038086
P7	2926	7000	500	1588215973.8111272	1588215973.9437616	0.13263440132141113
P8	2927	8000	500	1588215974.0814064	1588215974.2143242	0.1329178810119629
P9	2928	9000	500	1588215974.3466206	1588215974.4793682	0.13274765014648438

(其中time interval為end time - start time)

TIME_MEASUREMENT

Name	PID	Ready	Run	Start Time	End Time	Time interval
P0	2919	0	500	1588215971.7295628	1588215971.9049873	0.17542457580566406
P1	2920	1000	500	1588215972.0397592	1588215972.2168422	0.17708301544189453
P2	2921	2000	500	1588215972.3606958	1588215972.498988	0.1382920742034912
P3	2922	3000	500	1588215972.6368465	1588215972.7751346	0.13828802108764648
P4	2923	4000	500	1588215972.9172885	1588215973.0556314	0.13834285736083984
P5	2924	5000	500	1588215973.190481	1588215973.3636687	0.1731877326965332
P6	2925	6000	500	1588215973.5383532	1588215973.6771257	0.13877248764038086
P7	2926	7000	500	1588215973.8111272	1588215973.9437616	0.13263440132141113
P8	2927	8000	500	1588215974.0814064	1588215974.2143242	0.1329178810119629
P9	2928	9000	500	1588215974.3466206	1588215974.4793682	0.13274765014648438

平均500個unit time是 0.14776906967163086 s = 147769069.67163086 ns

因此之後我們考慮一個unit time = **295538.1393432617 ns**

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P0	2919	0	500	1588215971.7295628	1588215971.9049873	0.17542457580566406	593.5767755575936
P1	2920	1000	500	1588215972.0397592	1588215972.2168422	0.17708301544189453	599.1883681591976
P2	2921	2000	500	1588215972.3606958	1588215972.498988	0.1382920742034912	467.9330881313687
P3	2922	3000	500	1588215972.6368465	1588215972.7751346	0.13828802108764648	467.91937377337166
P4	2923	4000	500	1588215972.9172885	1588215973.0556314	0.13834285736083984	468.1049209698019
P5	2924	5000	500	1588215973.190481	1588215973.3636687	0.1731877326965332	586.0080633971207
P6	2925	6000	500	1588215973.5383532	1588215973.6771257	0.13877248764038086	469.55864291748605
P7	2926	7000	500	1588215973.8111272	1588215973.9437616	0.13263440132141113	448.7894578214113
P8	2927	8000	500	1588215974.0814064	1588215974.2143242	0.1329178810119629	449.74865615426165
P9	2928	9000	500	1588215974.3466206	1588215974.4793682	0.13274765014648438	449.17265311838685

a. First in, first served (FCFS)

- FIFO_1.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2940	0	500	1588215979.684374	1588215979.8539634	0.16958928108215332	573.8321336765903
P2	2941	0	500	1588215979.8541708	1588215979.9919147	0.13774394989013672	466.0784228940071
P3	2942	0	500	1588215979.9920523	1588215980.1312144	0.1391620635986328	470.8768347390819
P4	2943	0	500	1588215980.131244	1588215980.2814977	0.1502537727355957	508.40738548833764
P5	2944	0	500	1588215980.2816355	1588215980.4137387	0.13210320472717285	446.9920701968607

- 理論值(Unit time): 500 500 500 500 500
- 實際值(Unit time): 574 466 470 508 446

- FIFO_2.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2949	0	80000	1588215980.418779	1588216001.7031658	21.28438687324524	72019.08667538792
P2	2953	100	5000	1588216001.7034357	1588216003.02456	1.3211243152618408	4470.232905294774
P3	2954	200	1000	1588216003.024733	1588216003.2928832	0.26815009117126465	907.3282107248216
P4	2955	300	1000	1588216003.2931607	1588216003.581068	0.28790736198425293	974.1800588717052

- 理論值(Unit time): 80000 5000 1000 1000

- 實際值(Unit time): 72019 4470 907 974

- FIFO_3.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2960	0	8000	1588216003.5856502	1588216005.758363	2.172712802886963	7351.71713442846
P2	2961	200	5000	1588216005.7586129	1588216007.1450224	1.3864095211029053	4691.135716641357
P3	2962	300	3000	1588216007.145154	1588216007.962281	0.817126989364624	2764.8783036274963
P4	2963	400	1000	1588216007.9625099	1588216008.231823	0.26931309700012207	911.2634247430252
P5	2964	500	1000	1588216008.2320268	1588216008.5064898	0.2744629383087158	928.6887266686501
P6	2965	500	1000	1588216008.5066812	1588216008.7752864	0.26860523223876953	908.8682524551928
P7	2966	600	4000	1588216008.7753124	1588216009.857186	1.0818736553192139	3660.6904872695263

- 理論值(Unit time): 8000 5000 3000 1000 1000 1000 4000
- 實際值(Unit time): 7352 4691 2765 911 928 908 3660

- FIFO_4.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2971	0	2000	1588216009.8613577	1588216010.3993235	0.5379657745361328	1820.2922158594772
P2	2972	500	500	1588216010.3993573	1588216010.5360768	0.1367194652557373	462.6119172285251
P3	2973	500	200	1588216010.5361435	1588216010.593608	0.0574643611907959	194.43974750092127
P4	2974	1500	500	1588216010.593839	1588216010.7260172	0.13217830657958984	447.2461891832761

- 理論值(Unit time): 2000 500 200 500
- 實際值(Unit time): 1820 463 194 447

- FIFO_5.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2979	0	8000	1588216010.7300732	1588216012.8701339	2.1400606632232666	7241.233459677529
P2	2980	200	5000	1588216012.8703952	1588216014.2089074	1.3385121822357178	4529.067501101989
P3	2981	200	3000	1588216014.2090273	1588216015.0058842	0.7968568801879883	2696.2911858305188
P4	2982	400	1000	1588216015.006037	1588216015.270258	0.2642209529876709	894.0333507371225
P5	2983	400	1000	1588216015.2702913	1588216015.538962	0.26867055892944336	909.0892956370271
P6	2984	600	1000	1588216015.5390446	1588216015.8029833	0.26393866539001465	893.0781860389772
P7	2985	600	4000	1588216015.8030112	1588216016.8587275	1.0557162761688232	3572.1828611184073

- 理論值(Unit time): 8000 5000 3000 1000 1000 1000 4000
- 實際值(Unit time): 7241 4529 2696 894 909 893 3572

b. Round Robin (RR)

- RR_1.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3039	0	500	1588216037.2728853	1588216037.4299445	0.1570591926574707	531.4345992922746
P2	3040	0	500	1588216037.430001	1588216037.5617363	0.13173532485961914	445.74729052689594
P3	3041	0	500	1588216037.5620022	1588216037.702316	0.14031386375427246	474.7741325910585
P4	3042	0	500	1588216037.702381	1588216037.834397	0.13201618194580078	446.6976148633953
P5	3043	0	500	1588216037.8345652	1588216037.9660418	0.13147664070129395	444.8719917959097

- RR_2.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3048	600	4000	1588216037.9704564	1588216039.9831727	2.012716293334961	6810.343659223051
P2	3049	800	5000	1588216038.1092534	1588216040.4014745	2.2922210693359375	7756.092240513052

- RR_3.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3070	1200	5000	1588216040.4058287	1588216046.0788198	5.672991037368774	19195.46170918978
P2	3073	2400	4000	1588216040.8439586	1588216046.2107022	5.366743564605713	18159.224987108504
P3	3076	3600	3000	1588216041.2595828	1588216045.5500984	4.290515661239624	14517.637793801885
P4	3077	4800	7000	1588216041.40341	1588216048.5849557	7.181545734405518	24299.894931883206
P5	3078	5200	6000	1588216041.5352216	1588216048.3209386	6.785717010498047	22960.545889532623
P6	3079	5800	5000	1588216041.6673903	1588216047.7937589	6.126368522644043	20729.5360803784

- RR_4.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3135	0	8000	1588216048.5893962	1588216054.7212873	6.1318910121917725	20748.22229651281
P2	3136	200	5000	1588216048.7276065	1588216053.9202504	5.192643880844116	17570.131193162055
P3	3137	300	3000	1588216048.8593717	1588216052.4605925	3.6012208461761475	12185.299853950155
P4	3138	400	1000	1588216048.9905908	1588216050.0661845	1.0755937099456787	3639.441299643556
P5	3139	500	1000	1588216049.1227555	1588216050.19826	1.0755045413970947	3639.1395837676214
P6	3140	500	1000	1588216049.2588613	1588216050.3300116	1.071150302886963	3624.4063296441172
P7	3141	600	4000	1588216049.4028435	1588216053.3882225	3.9853789806365967	13485.159612538731

- RR_5.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3185	0	8000	1588216054.7257955	1588216060.8918085	6.16601300239563	20863.679442855013
P2	3186	200	5000	1588216054.8638022	1588216060.086766	5.222963809967041	17672.723464976112
P3	3187	200	3000	1588216054.9960938	1588216058.604613	3.6085193157196045	12209.995379068081
P4	3188	400	1000	1588216055.1277497	1588216056.1940036	1.066253900527954	3607.8385784567768
P5	3189	400	1000	1588216055.264016	1588216056.326344	1.0623281002044678	3594.5550126462517
P6	3190	600	1000	1588216055.3962886	1588216056.4631073	1.0668187141418457	3609.7497145800085
P7	3191	600	4000	1588216055.5287445	1588216059.5502064	4.0214619636535645	13607.252087970668

c. Shortest Job First (SJF)

- SJF_1.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2993	0	7000	1588216018.7419279	1588216020.6441057	1.9021778106689453	6436.319234112803
P2	2990	0	2000	1588216016.8633523	1588216017.3979027	0.5345504283905029	1808.7358524296358
P3	2991	100	1000	1588216017.3979774	1588216017.672559	0.2745816707611084	929.0904766852688
P4	2992	200	4000	1588216017.6725824	1588216018.7418063	1.0692238807678223	3617.887975960828

- SJF_2.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	2998	100	100	1588216020.6792061	1588216020.7098796	0.030673503875732422	103.78864786756255
P2	3001	100	4000	1588216022.6385171	1588216023.7029085	1.0643913745880127	3601.5364275936754
P3	2999	200	200	1588216020.7101574	1588216020.7742696	0.06411218643188477	216.93371479685646
P4	3002	200	4000	1588216023.7031145	1588216024.7667234	1.0636088848114014	3598.88874977331
P5	3000	200	7000	1588216020.7744832	1588216022.6384385	1.8639552593231201	6306.987191112322

- SJF_3.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3007	100	3000	1588216024.7993865	1588216025.6090367	0.809650182723999	2739.5793467577
P2	3014	100	5000	1588216029.643182	1588216030.9805415	1.3373594284057617	4525.166976342249
P3	3012	100	7000	1588216027.7763789	1588216029.6431491	1.8667702674865723	6316.512216104722
P4	3008	200	10	1588216025.6091204	1588216025.611913	0.0027925968170166016	9.449192659946524
P5	3009	200	10	1588216025.6120832	1588216025.615003	0.002919912338256836	9.87998484644114
P6	3010	300	4000	1588216025.615092	1588216026.6848469	1.0697548389434814	3619.6845568584376
P7	3011	400	4000	1588216026.6849916	1588216027.7763662	1.0913746356964111	3692.838555868422
P8	3017	500	9000	1588216030.9805963	1588216033.3640196	2.383423328399658	8064.689497254224

- SJF_4.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3022	0	3000	1588216033.3683627	1588216034.1850903	0.8167276382446289	2763.527036001319
P2	3023	1000	1000	1588216034.1852653	1588216034.4491725	0.26390719413757324	892.971698082765
P3	3024	2000	4000	1588216034.4493616	1588216035.5145075	1.0651459693908691	3604.089718361944
P4	3026	5000	2000	1588216035.7785442	1588216036.310963	0.5324187278747559	1801.5229068501444
P5	3025	7000	1000	1588216035.5145223	1588216035.778394	0.2638716697692871	892.8514957685559

- SJF_5.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3031	0	2000	1588216036.315344	1588216036.8673203	0.551976203918457	1867.698717820469
P2	3032	500	500	1588216036.8675056	1588216036.999545	0.13203954696655273	446.7766741036134
P3	3033	1000	500	1588216036.9995925	1588216037.1363893	0.1367967128753662	462.8732967574095
P4	3034	1500	500	1588216037.1367118	1588216037.2683852	0.13167333602905273	445.53754152223564

d. Preemptive Shortest Job First (PSJF)

- PSJF_1.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3235	0	10000	1588216060.896367	1588216067.6231964	6.726829290390015	22761.289982193925
P2	3241	1000	7000	1588216063.5867143	1588216067.0851336	3.4984192848205566	11837.454524802335
P3	3236	2000	5000	1588216061.4365566	1588216063.5864654	2.1499087810516357	7274.556122702523
P4	3237	3000	3000	1588216061.9744995	1588216063.3151443	1.3406448364257812	4536.283673589244

- PSJF_2.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3253	0	3000	1588216067.6277776	1588216068.4428942	0.8151166439056396	2758.075982060974
P2	3255	1000	1000	1588216068.443129	1588216068.7115676	0.26843857765197754	908.3043503234329
P3	3256	2000	4000	1588216068.7118084	1588216069.7941039	1.0822954177856445	3662.1175872281574
P4	3259	5000	2000	1588216070.0685382	1588216070.6117904	0.5432522296905518	1838.1797723222958
P5	3258	7000	1000	1588216069.7941961	1588216070.0682755	0.2740793228149414	927.3907030205793

- PSJF_3.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3264	0	2000	1588216070.616197	1588216071.1776736	0.561476469039917	1899.844366238542
P2	3265	500	500	1588216071.177815	1588216071.3195405	0.1417255401611328	479.55076280872635
P3	3266	1000	500	1588216071.3197486	1588216071.4540424	0.134293794631958	454.40427733077934
P4	3267	1500	500	1588216071.4543078	1588216071.5881836	0.13387584686279297	452.99008500320434

- PSJF_4.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3275	0	7000	1588216072.950093	1588216075.3589306	2.4088375568389893	8150.68255530242
P2	3272	0	2000	1588216071.593016	1588216072.1467319	0.5537159442901611	1873.5854043089544
P3	3273	100	1000	1588216072.1468494	1588216072.4150004	0.26815104484558105	907.3314376325856
P4	3274	200	4000	1588216072.415249	1588216074.0216622	1.6064131259918213	5435.552682173465

- PSJF_5.txt

Name	PID	Ready	Run	Start Time	End Time	Time Interval	Unit Time
P1	3284	100	100	1588216075.3935757	1588216075.4211242	0.027548551559448242	93.21487785186041
P2	3287	100	4000	1588216076.028108	1588216077.6867435	1.6586356163024902	5612.255731149534
P3	3285	200	200	1588216075.421287	1588216075.4838479	0.0625607967376709	211.68434259176198
P4	3288	200	4000	1588216076.575944	1588216078.2263236	1.6503796577453613	5584.320390636546
P5	3286	200	7000	1588216075.484078	1588216079.5780482	4.09397029876709	13852.595498721821

4. Conclusion

- Q1. 為什麼各個結果所耗費的unit time會比理論值小？
 - A1.1 可能因為當初生成的unit time比理論上還大一些
 - A1.2 此外，排程器上的counter基本上無法和child process上的counter同步，scheduler 還要忙signal handle, system call, heap maintain 之類的是，當counter跑到一定時間後，這些差異就會很明顯了
- Q2. 為什麼生成的unit time會比理論值大？
 - A2.1 除了排程外，系統也同時在做別的事
 - A2.2 VM上仍無可避免有其他程式在運行，當他們跟我們的行程context switch時，也會產生許多overhead
 - A2.3 同時，CPU也在處理I/O，這些interrupt會讓overhead更大