作业 5：

5.1 现有 5 个作业要在一台计算机上依次执行，它们的运行时间分别是 9，3，5，11 和 X。请问：1）该以何种顺序运行这 5 个作业，从而可以获得最短的平均响应时间？）如果要获得最短的平均周转时间，该以何种顺序运行这 5 个作业？

解答：

X 的时间不确定,为了获得最短的平均响应时间,可以按照最短作业优先(Shortest Time First Complete，STFC）的方式运行这 5 个作业。来获得最短响应时间。

X 的值	排序	最短响应时间	最短周转时间
X<3	X,3,5,9,11	$\frac{0+X+X+3+X+8+X+17}{5}$	$\frac{X+X+3+X+8+X+17+X+28}{5}$
3<=X<5	3,X,5,9,11	$\frac{0+3+X+3+X+8+X+17}{5}$	$\frac{3+X+3+X+8+X+17+X+28}{5}$
5<=X<9	3,5,X,9,11	$\frac{0+3+8+X+8+X+17}{5}$	$\frac{3+8+X+8+X+17+X+28}{5}$
9<=X<11	3,5,9,X,11	$\frac{0+3+8+17+X+17}{5}$	$\frac{3+8+17+17+X+X+28}{5}$
X>11	3,5,9,11,X	$\frac{0+3+8+17+28}{5}$	$\frac{3+8+17+28+X+28}{5}$

5.2 现有 5 个作业（作业 A、B、C、D、E）要在一台计算机上执行。假设它们在同一时间被提交，同时它们的运行时间分别是 10、8、4、12 和 15 分钟。当使用以下 CPU 调度算法运行这 5 个作业时，请计算平均等待时间。

（1）Round robin 算法（使用该算法时，每个作业分到的 CPU 时间片相等)

（2）优先级调度算法（作业 A-E 的优先级分别是：2,5,1,3,4，其中 5 是最高优先级，1 是最低优先级）

（3）First-come，first-served 算法（假设作业的达到顺序是 A，B，C，D，E)

（4）Shortest job first 算法

注意：假设作业切换可以瞬时完成，即开销为 0。

解答：

（1） 选择时间片为 4

假设按照 STFC 将任务放入队列

甘特图如下：

C	B	A	D	E	B	A	D	E	A	D	E	E

0 4 8 12 16 20 24 28 32 36 38 42 46 49

则等待时间分别为：

C:0

B:4+12=16

A:8+12+12=32

D:12+12+6=30

E:16+12+6=34

平均等待时间为：$\frac{16+32+30+34}{5} = 22.4$

（2）优先调度算法：考虑最高优先级优先调度，并且固定优先级

调度顺序为：B,E,D,A,C

等待时间分别为：

A:8+12+15=35

B:0

C:10+8+12+15=45

D:8+15=23

E:8

平均等待时间为：$\frac{35+45+23+8}{5} = 22.2$

（3）FCFS 算法，

调度顺序为 A,B,C,D,E

等待时间分别为：

A:0

B:10

C:10+8=18

D:18+4=22

E:22+12=34

平均等待时间为：$\frac{10+18+22+34}{5} = 16.8$

（4）SJFC 算法

调度顺序为 C,B,A,D,E

等待时机分别为：

A:4+8=12

B:4

C:0

D:10+12=22

E:22+12=34

平均等待时间为：$\frac{4+12+22+34}{5} = 14.4$

5.3 A real-time system needs to handle two voice calls that each run every 5 msec and consume 1 msec of CPU time per burst, plus one video at 24 frames/sec, with each frame requiring 20 msec of CPU time. Is this system schedulable?

Answer:

This is a Periodic scheduling issue, processes can be scheduled only if the system can guarantee real-time for all processes, and the conditions under which scheduling can be performed are $\sum \frac{C_i}{T_i} \leq 1$.

As for this question, we have $\frac{0.001}{0.005} * 2 + \frac{0.02}{\frac{1}{24}} = 0.88 \leq 1$, so this system is schedulable.

5.4 作为容器技术的重要基础，cgroups 为 Linux 提供了内存、CPU 等资源的分配与限制功能。Cgroups 的使用手册可在 Linux 系统中通过`man cgroups`指令查看，或访问官方网页[1]。请重点关注 cgroups 中 cpu 子系统的文档[2]及其基本用法，回答以下问题，并附上必要代码和截图：

（1） 回顾上一次作业中的绑核操作，写一个简单的程序，使其绑定 1 号 CPU，且 CPU 占用率达到 100%

（2） 应用 cgroups 功能，将（1）中程序的 CPU 占用率限制在 30%以下

（3） 重新启动共计 2 个（1）中的程序，观察它们各自的 CPU 占用率

（4） 应用 cgroups 功能，将（3）中程序的 CPU 占用率调整为 2:1，并验证你的实现效果

注 1：强烈建议同学优先尝试阅读官方手册，学习从说明文档获取关键信息的能力；若确实存在困难，可查询中文资料并注明参考出处

注 2：可以通过 Linux 的`top`命令查看系统内各进程的 CPU 等资源占用率

[1] https://man7.org/linux/man-pages/man7/cgroups.7.html

[2] https://www.kernel.org/doc/Documentation/scheduler/sched-bwc.txt

(1)

代码：

```c
#define _GNU_SOURCE
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(1, &cpuset);
    sched_setaffinity(0, sizeof(cpu_set_t), &cpuset);
    while (1) {
    }
    return 0;
}
```

首先利用 cat 命令查看 CPU 相关信息，第一行表示物理 CPU 为 1，第二个表示 CPU 核数为 8，最后一行报名逻辑 CPU 为 16.

```
solomon@DESKTOP-23PER74:~/CODES/C/OS$ cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l
1
solomon@DESKTOP-23PER74:~/CODES/C/OS$ cat /proc/cpuinfo| grep "cpu cores"| uniq
cpu cores       : 8
solomon@DESKTOP-23PER74:~/CODES/C/OS$ cat /proc/cpuinfo| grep "processor"| wc -l
16
```

运行该程序，并使用 top 命令查看各个进程资源占用情况，如下图：

```
top - 18:47:12 up 21 min,  0 users,  load average: 0.26, 0.13, 0.06
Tasks:  21 total,   2 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.4 us,  0.0 sy,  0.0 ni, 93.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7826.9 total,   7116.0 free,    410.3 used,    300.5 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7191.2 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3555	solomon	20	0	2364	516	448	R	100.0	0.0	0:14.02	cgroup
218	solomon	20	0	11.0g	152904	48904	S	0.7	1.9	0:18.57	node
724	solomon	20	0	662436	65536	39400	S	0.7	0.8	0:01.63	node
43	solomon	20	0	960996	103288	45020	S	0.3	1.3	0:06.37	node
1915	solomon	20	0	600228	54428	37260	S	0.3	0.7	0:00.37	node
1	root	20	0	1804	1188	1104	S	0.0	0.0	0:00.01	init
11	root	20	0	1824	88	0	S	0.0	0.0	0:00.00	init
12	root	20	0	1824	104	0	S	0.0	0.0	0:00.00	init
13	solomon	20	0	2612	524	456	S	0.0	0.0	0:00.00	sh
14	solomon	20	0	2612	592	520	S	0.0	0.0	0:00.00	sh
39	solomon	20	0	2612	592	524	S	0.0	0.0	0:00.00	sh
196	solomon	20	0	849560	54308	38864	S	0.0	0.7	0:00.14	node
735	solomon	20	0	10032	5292	3540	S	0.0	0.1	0:00.02	bash

其中，各参数分别为：

- PID — 进程 id
- USER — 进程所有者
- PR — 进程优先级
- NI — nice 值。负值表示高优先级，正值表示低优先级
- VIRT — 进程使用的虚拟内存总量，单位 kb。VIRT=SWAP+RES
- RES — 进程使用的、未被换出的物理内存大小，单位 kb。RES=CODE+DATA
- SHR — 共享内存大小，单位 kb
- S —进程状态。D=不可中断的睡眠状态 R=运行 S=睡眠 T=跟踪/停止 Z=僵尸进程
- %CPU — 上次更新到现在的 CPU 时间占用百分比
- %MEM — 进程使用的物理内存百分比
- TIME+ — 进程使用的 CPU 时间总计，单位 1/100 秒
- COMMAND — 进程名称（命令名/命令行）

然后输入“1”可以查看各个 CPU 使用情况，结果如下图：

```
Tasks:  21 total,   2 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  0.3 us,  0.7 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

可以发现 CPU1 的使用率变为 100%，其他核均处于 sleep 态。

（2） Cgroup 简介

Linux Cgroup 可为系统中所运行任务(进程)的用户定义组群分配资源 —— 比如 CPU 时间、系统内存、网络带宽或者这些资源的组合。可以通过监控您配置的 cgroup，拒绝 cgroup 访问某些资源，甚至在运行的系统中动态配置 cgroup。

所以，可以将 controll groups 理解为 controller （system resource） （for）（process）groups，也就是是说它以一组进程为目标进行系统资源分配和控制。

在 Cgroups 中,CPU 资源的控制也有两种策略,一种是完全公平调度（CFS:Completely Fair Scheduler）策略，提供了限额和按比例分配两种方式进行资源控制；另一种是实时调度（Real-Time Scheduler）策略，针对实时进程按周期分配固定的运行时间。配置时间都以微秒（μs）为单位，文件名中用 us 表示。

在 CFS 调度策略下的配置其中两个比较重要的参数为：

- cpu.cfs_period_us：设定周期时间，必须与 cfs_quota_us 配合使用。
- cpu.cfs_quota_us ：设定周期内最多可使用的时间。这里的配置指 task 对单个 cpu 的使用上限，若 cfs_quota_us 是 cfs_period_us 的两倍，就表示在两个核上完全使用。数值范围为 1000 - 1000,000（微秒）

可以通过修改其设置来达到使用率为 30%的设置，

具体过程如下：

测试程序的资源使用率：

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
11474 solomon   20   0    2364    580    512 R 100.0   0.0   0:22.62 cgroup
```

```
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ sudo sh -c "echo 100000 > cpu.cfs_period_us"
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ sudo sh -c "echo 30000 > cpu.cfs_quota_us"
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ cat cpu.cfs_quota_us
30000
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ echo 11474 > tasks
bash: tasks: Permission denied
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ sudo echo 11474 > tasks
bash: tasks: Permission denied
solomon@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test$ sudo sh -c "echo 11474 > tasks"
```

注意：sudo 写不了，是因为“>”也是一个命令，sudo 只是让 echo 具有 root 权限，而“>”还是普通权限，此时利用 sh -c 命令让 bash 把字符串当成完整一个命令执行，如：sudo sh -c “command”.

通过资源控制后，测试进程最高占用 30%的 CPU 资源，也就是我们设置的赋值。

（3）

如果同时启动两个测试程序，那么两个测试程序共同占用 30%的 CPU 资源。因为同样的进程共同占用所分配的资料，

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
11474 solomon   20   0    2364    580    512 R  30.0   0.0   5:11.78 cgroup
```

```
top - 19:55:45 up  1:29,  0 users,  load average: 0.50, 0.58, 0.29
Tasks:  22 total,   3 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  : 30.6 us,  0.0 sy,  0.0 ni, 69.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

启动两个测试程序的结果：

```
solomon@DESKTOP-23PER74:~/CODES/C/OS/homework5$ ./cgroup fork & ./cgroup
[1] 16269
```

将进程号加入 cgroup

```
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test# cat cpu.cfs_quota_us
30000
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test#  echo 16269 > tasks
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test# echo 16270 > tasks
```

利用 top 命令查看结果

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
16270 solomon   20   0    2364    580    508 R  15.3   0.0   1:25.72 cgroup
16269 solomon   20   0    2364    512    444 R  14.6   0.0   1:17.77 cgroup

Tasks:  25 total,   3 running,  22 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.0 us,  0.0 sy,  0.0 ni, 97.7 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu1  : 31.0 us,  0.0 sy,  0.0 ni, 69.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

可以发现两个进程均在 cpu1 中运行，且总利用率为 30%，但每个进程的利用率平均为 15%。

(4)将两个进程分别加入两个不同的 cgroup，然后分别配置其 cpu 的使用时间，即可实现两个进程的资源占用为 2：1

将两个进程分别加入不同的 Cgroup 即可，其中新建 test2 的 Cgroup 中，cpu.cfs_quota_us 为 test 中的一半，将其中一个运行进程加入 test2,另一个加入 test.则可实现两个进程的利用率为 2：1

```
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test1# sudo echo 100000 > cpu.cfs_period_us
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test1# sudo echo 15000 > cpu.cfs_quota_us
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test1# cat cpu.cfs_quota_us
15000
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test1# echo 19958 > tasks
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test1# cd ..
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu# cd test
root@DESKTOP-23PER74:/sys/fs/cgroup/cpu/test# echo 19959 > tasks
```

可以看到此时两个进程的占用率分别为 30%和 15%，满足要求，但是 cpu1 的利用率达到了 45%。

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
19959 solomon   20   0    2364    512    448 R  30.0   0.0   1:11.27 cgroup
19958 solomon   20   0    2364    584    512 R  15.3   0.0   0:38.57 cgroup

Tasks:  28 total,   3 running,  25 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  : 45.7 us,  0.0 sy,  0.0 ni, 54.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

参考链接：https://blog.csdn.net/lisemi/article/details/94021498