作业 5

毕定钧 2021K8009906014

本次作业包含:

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？

2）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？

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。

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*?

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

5.1

(1)

为了获得最短的平均响应时间，应当使用最短时间优先算法（$STCF$）。由于 X 未知，它可以很小也可以很大，因此最好将其放在最后。排序如下：$3-5-9-11-X$。

$$\text{平均响应时间} = \frac{0+3+8+17+28}{5} = 11.2。$$

(2)

为了获得最短的平均周转时间，可以使用先到先服务算法（FCFS）并按如下顺序安排：$3-5-9-11-X$。

$$\text{平均周转时间} = \frac{3+8+17+28+28+X}{5} = \frac{84+X}{5}。$$

5.2

(1)

将时间片分别设置为 1、3、5、8、10、15，可以得出其调度顺序如下表所示：

时间片	执行顺序
1	ABCDEABCDEABCDEABCDEABDEABDEABDEABDEADEADEDEDEEEE
3	ABCDEABCDEABDEADEE
5	ABCDEABDEDE
8	ABCDEADE
10	ABCDEDE
15	ABCDE

各种时间段划分的平均等待时间（Average Wait Time）如下：

时间片	A	B	C	D	E	AWT
1	30	26	14	33	34	27.4
3	30	25	18	31	34	27.6
5	19	19	10	32	34	22.8
8	28	8	16	30	34	23.2
10	0	10	18	32	34	18.8
15	0	10	18	22	34	16.8

(2)

不考虑优先级反转与优先级继承，按 $B-E-D-A-C$ 的顺序执行:

$$AWT = \frac{0+8+23+35+45}{5} = 22.2。$$

(3)

$$AWT = \frac{0+10+18+22+34}{5} = 16.8。$$

(4)

$$AWT = \frac{0+4+12+22+34}{5} = 14.4。$$

5.3

每个语音呼叫的利用率为 0.2，语音呼叫的总利用率为 0.4。视频流的利用率为 0.48，故总利用率为 0.88。根据利用率上限 $U = 3 \times (2^{1/3} - 1) = 2.897$，可知系统是可调度的。

5.4

(1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sched.h>
#include <unistd.h>

int main() {
    int num_cores = sysconf(_SC_NPROCESSORS_ONLN);
    cpu_set_t cpu_set;

    CPU_ZERO(&cpu_set);
    CPU_SET(0, &cpu_set);
    sched_setaffinity(0, sizeof(cpu_set), &cpu_set);

    struct sched_param sp;
    sp.sched_priority = 99;
    if (sched_setscheduler(0, SCHED_RR, &sp) == -1) {
        perror("sched_setscheduler");
        exit(1);
    }
    while (1);
    return 0;
}
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4605 root      rt   0    2356    576    512 R  96.0   0.0   0:24.29 a.out
```

(2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sched.h>
#include <unistd.h>
#include <cgroup.h>

int main() {
    struct cgroup *cg;
    cg = cgroup_new_cgroup("my_limit_group");
    struct cgroup_controller *cpu_c = cgroup_add_controller(cg, "cpu");

    cgroup_set_value_uint64(cpu_c, "cfs_quota_us", 30000);
    cgroup_set_value_uint64(cpu_c, "cfs_period_us", 100000);
    cgroup_create_cgroup(cg, 0);

    cpu_set_t cpu_set;
    CPU_ZERO(&cpu_set);
    CPU_SET(0, &cpu_set);
    sched_setaffinity(0, sizeof(cpu_set), &cpu_set);

    while (1);
    cgroup_delete_cgroup(cg, 0);
    return 0;
}
```

(3)

将 (1) 的不限制 CPU 占用率的程序记为程序 A，(2) 中限制为 30% 的程序记为程序 B，同时运行时程序 A 的占用率略高于 70%，程序 B 约高于 25% 但低于 28%。

(4)

占用率分别为 60% 和 30%。