《操作系统实现》上机实验

进程调度

20201016

Xinu 采用了基于优先级的抢占式调度算法。其它系统可能采用诸如先到先服务、最短作业优先、最短剩余时间优先、纯粹的时间片轮转（xv6 系统）等各种调度算法。感兴趣的同学可以查阅资料，了解 Linux 中使用的完全公平调度算法（Completely Fair Scheduler，CFS）。虽然使用了优先级，但 CFS 会保证让每个进程在一个调度周期内至少有机会运行一次。

本次上机实验将根据要求在 Xinu 中实现一种调度算法，使得高优先级进程拥有较多的时间片，但低优先级的进程仍然能够执行（即使高优先级进程永不停止）。

允许修改任意代码和数据结构。但请注意，在我的机器上（QEMU 2.12，Ubuntu 18.04），如下的忙等待代码（1），在 2ms 的时间片内，大约会循环 60000 次；而包含数组操作的代码(2)，大约会循环 2w 次。在必要时，请根据自己的环境做出调整和说明。

本次实验基于原 Xinu 的抢占式调度，而不是基于上次上机实验的修改。

本次实验预计占用 3 次上机课时间（10/16，10/23，10/30），即截止日期为 11 月 6 日 0 点。

```
while (counter ++ < 0xFFFFFFFF);
```

(1)

```
while (counter ++ < 0xFFFFFFFF){
    array[idx++] = counter;
    idx = idx % 1000;
}
```

(2)

1. 实现如下要求的调度算法

a) 假设系统中除了空进程之外，存在 3 种有优先级：P1>P2>P3。每种优先级所对应的初始时间片为 T1 （30ms）、T2（20ms）、T3（10ms）

b) 每个进程在自己的时间片耗尽之后，无论优先级高低，都需要让出 CPU，直到当前系统中所有可运行的进程都耗尽了时间片，才能再次被调度。假设 3 个进程 A、B、C 分别是高中低优先级进程，进程 A 执行完 30ms 后，进程 B 执行 20ms，然后进程 C 执行 10ms，接下来重复这个过程

i. 在实验报告中详细描述实现方法，并说明有何优劣

c) 如果某个进程执行 N 毫秒之后，因为等待信号量而暂停或者执行 sleepms(X)从而睡眠 X 毫秒，当从等待或睡眠中恢复后，此轮可用时间片为（初始时间片 - N）

d) 进程本身不处理时间片的变化

e) 在实验报告中描述调度算法的实现思路、对应于代码的更改、新增或改变的各变量/数据结构/函数的用途

f) 修改 main.c，在 main 函数中使用延迟调度的方式创建具有三种优先级的进程若干，每个进程内可适当使用前述的忙等待（根据自己的实验环境选择循环次数，应该确保一个完整的循环不会耗尽一个普通的 10ms 时间片）、等待信号量、睡眠等方式

i. 当使用信号量时，需确保不出现死锁（如：所有普通进程都在等待，系统只有一个空进程在运行）

g) 在运行一段时间后，通过调试器等方式查看各个进程的总 CPU 运行时间

h) 将完整的代码打包，文件名为 xinu-1016-1，打包格式可为 zip 或 tar.gz 等，打包前请注意执行 make clean 命令清除编译产生的二进制文件

2. 生产者-消费者交互：在上一题的基础上，假设系统中存在三种进程，生产者、消费者和摇摆者，每种类型有若干进程（三者总数量不超过 30）。请实现如下要求的进程调度算法和生产者消费者交互。

a) 如果初始时生产者是高优先级，则消费者是低优先级；反之亦然。摇摆者永远处于中优先级

b) 存在一个环形缓冲区（大小自定），每个消费者向其中写入非零数据，消费者读取非零数据之后将相应位置置零

   i. 消费者和生产者通过前述的忙等待、等待信号量、睡眠等方式，确保当前进程在不超过一个普通 10ms 时间片的范围内，读/写缓冲区中固定数目的数据槽，该数目较小（考虑前述的 2ms 时间片会操作约 2w 个数据）

   ii. 在最多 10ms 的时间片周期内，必然会发生一次调度（通过时间片轮转、等待信号量、睡眠等方式），因此即使高优先级进程拥有 30ms 的时间片，至少需要被调度 3 次才能用尽时间片

c) 当某个摇摆者进程即将被调度时，检查当前系统中消费者进程和生产者进程的总期望运行时间 CT 和 PT，如果 CT < PT，则待调度的摇摆者进程将执行消费者进程的操作，否则执行生产者进程的操作

   i. 一个调度周期满足如下条件：所有进程都至少被执行一次；如果最后一个可用进程因任何原因（时间片耗尽、睡眠、等待信号量）交出 CPU 控制权，所有进程要么耗尽初始时间片，要么在等待信号量或睡眠（且在此刻无法恢复）

   ii. 总期望运行时间的计算方法（以消费者进程为例）：在一个调度周期内，已经执行过的消费者进程（包含耗尽时间片、在等待、在睡眠的消费者进程以及按消费者方式执行的摇摆者进程）在 CPU 上执行的总时间 + 当前调度周期内

待执行的所有消费者进程的期望执行时间（尚未执行过的进程，以各进程分配的初始时间片为准，如 10ms 或 30ms；已调度执行过的进程，则以剩余时间片为准）之和

iii. 如：每个消费者进程的初始时间片为 10ms，在调度某个摇摆者进程时，生产者进程已经完全执行完毕，使用 CPU 共 134ms（某些生产者阻塞了），已执行的消费者进程共占用 CPU 70ms，就绪队列中的消费者期望运行总时间 50ms，则目标摇摆者将被作为消费者执行

iv. 摇摆者执行消费者操作或生产者操作过程中（因等待信号量、睡眠等），如果发生了重新调度，则该进程再次调度回来时，仍然继续此前的操作，直到一个不到 10ms 的消费者/生产者周期执行完毕

v. 考虑结合"消息传递"的机制实现摇摆者的功能切换。如果有其他的可行的实现方法，亦可。在实验报告中说明实现策略

d) 各种进程自身并不计算时间片等信息，由调度器等组件处理

e) 系统中已知只存在这三种进程（不含空进程），当一个调度周期结束后，消费者与生产者的优先级/初始时间片互换（摇摆者保持不变）

i. 注意：一个调度周期结束后，从睡眠或等待中恢复的，前一轮尚未耗尽时间片的消费者/生产者进程在当前周期内已占用时间片清零，新的优先级/初始时间片根据当前周期

ii. 在上一个调度周期因等待信号量或睡眠而阻塞的摇摆者进程，进入当前周期后被重新调度时，已使用时间片清零，继续执行上一个周期未完成的操作，直到该消费/生产周期结束，然后根据前述的摇摆算法，选择作为消费者还是生产者（参考 c）

iii. 睡眠进程结束睡眠进入调度器时，状态为 PR_SLEEP

f) 修改 main.c，创建三种进程所对应的实现，并在 main 函数中使用延迟调度的方式创建三种进程若干。在运行一段时间后通过调试器等方式查看各种进程的总运行时间

g) 将完整的代码打包，文件名为 xinu-1016-2，打包格式可为 zip 或 tar.gz 等，打包前请注意执行 make clean 命令清除编译产生的二进制文件

3. 最终提交一份实验报告和两个压缩包文件，整体打包成一个压缩包，提交到 OBE 系统，截止日期：2020 年 11 月 6 日 0 点。

a) 由于不同机器、不同版本的 QEMU 执行效率不同，在实验报告中给出观察结果或解释所选用的忙等待循环次数等信息时，阐明实验环境