Project 2: The HOST Dispatcher Shell

- 本次课程项目的负责助教为付昊源
  - Email: simon-fuhaoyuan@sjtu.edu.cn
  - WeChat: 17321006287
- 在完成课程项目的过程中有任何问题，欢迎在Canvas对应的讨论区进行提问或与助教取得联系。
  - **提问时请优先在QQ群中进行，这样可以方便其他同学进行参考，谢谢！**

1. 基本功能

Hypothetical Operating System Testbed (以下简称HOST)是一个多进程系统。HOST中的进程受**四优先级调度器**控制，同时受制于**有限的软硬件资源**。

四优先级进程调度器

调度器采取四优先级的策略：

1. 实时进程（优先级为0）享有最高优先级，必须遵循**先进先出（FCFS）**的原则，且优先于其他低优先级的进程。实时进程一直运行至进程结束为止。
2. 用户进程（优先级为1，2或3）遵循**三级反馈调度器**调度。基本的时间片为1秒，同时反馈调度器的时间片也是1秒。

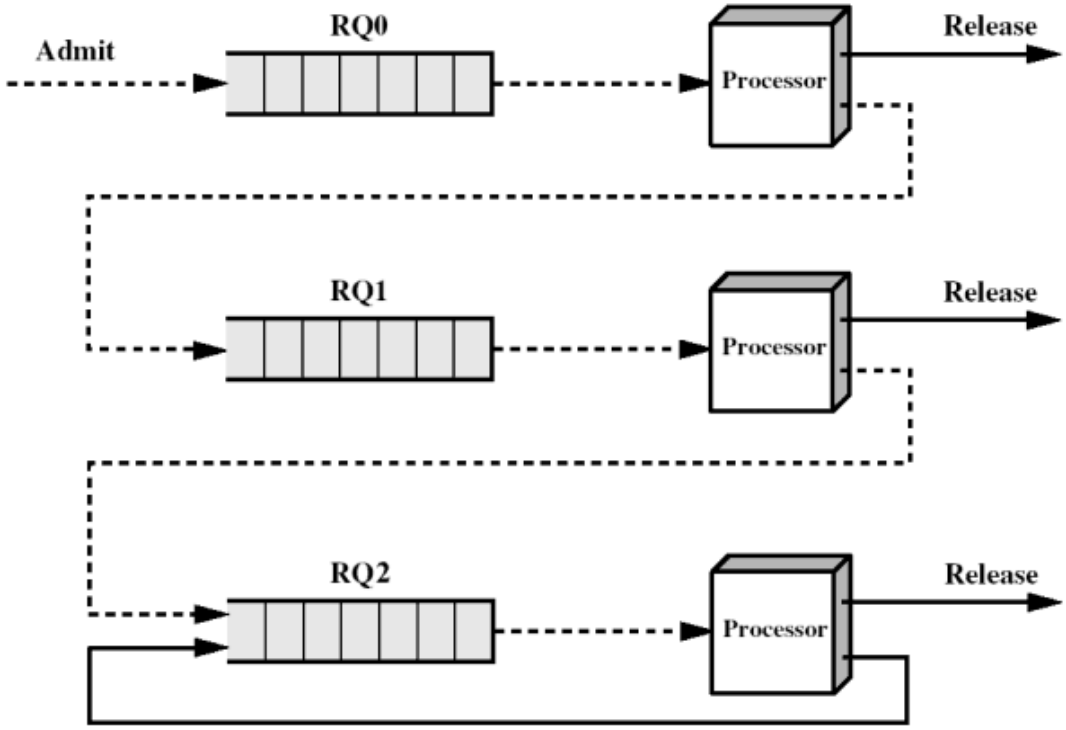

Figure 1. Three Level Feedback Dispatcher

HOST中的调度器需要维护两个提交队列：`实时进程队列`和`用户进程队列`，这两个队列中的进程是由`工作调度表`给出的。`工作调度表`需要在调度器的每一个时间片中检查已到达的进程，并将其传送到合适的提交队列中。调度器则需要保证提交队列中的实时进程都一直运行直至结束，如果此时有低优先级的进程在运行，则需暂停该进程，优先运行实时进程。在`用户进程队列`中的进程开始运行之前，`实时进程队列`中的进程必须为空。

前面提到，用户进程进一步分为1，2和3这三种优先级，因此又有三个`优先级队列`分别与之对应。所有已到达的用户进程，只要现有资源可以满足其运行所需资源（内存和I/O设备），它就会被传送到对应的`优先级队列`。标准的反馈队列会先接受所有最高优先级（即优先级为1）的用户进程，然后每一个时间片过后将该进程优先级下降一级。同时，HOST中的调度器支持接受多种优先级的用户进程，并将其插入对应的`优先级队列`，例如，有一些进程从`工作调度表`传送过来时，优先级即为2，则这样的进程无权进入优先级为1的`优先级队列`，而会直接被插入优先级为2的`优先级队列`；因此，HOST中的调度器需要在优先级为3的`优先级队列`中需要模拟Round Robin调度算法，直至进程被执行完毕。

当所有满足资源限制的高优先级进程结束后，调度器将会开始/继续执行当前非空且优先级最高的`优先级队列`中处于队首的进程。下一个时间片中，如果当前或更高优先级的`优先级队列`中有其他进程，则当前的进程则被挂起（若进程结束则释放其资源），转而执行其他进程。整体的进程逻辑如下：

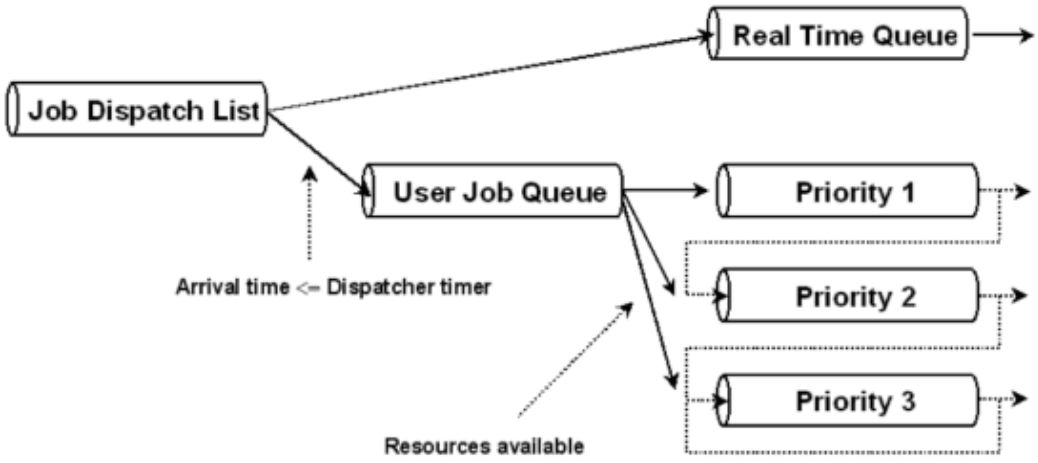

Figure 3. Dispatcher Logic Flow

资源限制

HOST系统中有如下的软硬件资源限制：

- 2个打印机（printer）
- 1个扫描仪（scanner）
- 1个调制解调器（modem）
- 2个光驱（CD driver）
- 1024Mb内存

用户进程可以使用任意资源，但进程在提交时（即在`工作调度表`中时），调度器会被告知该进程将会使用哪些资源。调度器需要保证，一旦某进程从`用户进程队列`传送到对应`优先级队列`，在该进程整个执行过程中（包括其挂起时间），进程需求的每一份资源只可用于该进程本身。

实时进程不会需求任何I/O资源，但是很显然它需要被分配内存。实时进程所需内存不会超过64Mb。

内存分配

为某进程分配内存时，必须保证分配的内存是一段**连续**的内存空间，且该段内存在进程的执行过程中始终属于该进程。为了保证实时进程不被阻塞，必须留有足够的连续空闲内存空间：即，任何时候都必须留有至少64Mb大小的连续内存空间，其他960Mb的内存即可用于其他用户进程。

HOST系统中的硬件MMU不能支持虚拟内存，不可以与磁盘进行交换，同时也不是一个页文件系统。为了满足上述限制，任何合适的内存分配算法都可以使用（First Fit, Next Fit, Best Fit, Worst Fit, Buddy等）。

进程

HOST中的进程是由你的Linux系统中某一个优先级很低的实际进程来进行模拟的，该进程每睡眠1秒，会输出以下信息：

1. 当进程开始时，它会输出进程ID；
2. 进程运行过程中，每秒钟会输出一个规则信息；
3. 当进程被挂起/继续/结束时，会输出一个提示信息。

如果20秒内，你的HOST调度器没有将该进程结束，该进程将会自动结束。进程的输出会伴随有一个随机生成的颜色框，从而不同的进程可以很明显地被区分出来。进程部分的代码无需手写，我们已经提供好了，在`sigtrap.c`中。**这一部分的代码请勿修改。**

进程的生命周期为：

1. 所有进程以txt文件的形式进行描述，作为输入提交到调度器中。该描述包括进程的到达时间、优先级、需要执行的时间（秒）、所需内存大小以及其他所需资源（工作调度表部分有详细描述）。
2. 如果进程已到达（即到达时间小于等于HOST已运行时间），且其所需资源都可用，则该进程从`工作调度表`传送到`实时进程队列`或`用户进程队列`。
3. `实时进程队列`中的进程由FCFS策略进行调度。
4. 如果足够的资源和内存可供用户进程使用，则用户进程获得该资源，并被传送到对应的`优先级队列`，服从三级反馈调度策略。同时，HOST中的剩余资源更新。
5. 当一个进程开始（`fork`和`exec("process", ...)`）时，调度器会输出进程的相关参数（进程ID，优先级，剩余执行时间，内存地址和大小以及系统为其分配的资源）。接下来将会执行`exec`。
6. 实时进程会持续执行，直到其被执行完，系统会向其发送`SIGINT`指令。
7. 用户进程可以执行一个时间片（1秒），然后将会被挂起（`SIGTSTP`）；如果其执行时间已到，则结束（`SIGINT`）。如果挂起，则该进程的优先级降1（最低为3），同时该进程被传送到修改优先级后的`优先级队列`中重新排队。
8. 如果没有实时进程在等待，则最高优先级的用户进程开始执行或继续执行（`SIGCONT`）。
9. 当一个进程结束后，它所占用的资源释放给HOST调度器，从而其他进程可以分配使用。
10. 当`工作调度表`、`用户进程队列`、`实时进程队列`以及三个`优先级队列`中都没有进程之后，调度器结束执行。

工作调度表

`工作调度表`是等待调度器进行调度的一个进程序列。这个序列由一个txt文件进行描述，且可以通过命令行的方式将该文件传给HOST调度器，示例如下：

```
./hostd <dispatchlist>
```

`工作调度表`（即该txt文件）中的每一行描述了一个进程的所需资源，包含以下信息：

```
<到达时间>, <优先级>, <执行时间>, <内存大小（Mb）>, <#打印机>, <#扫描仪>, <#调制解调器>, <#光驱>
```

例如，

```
12, 0, 1, 64, 0, 0, 0, 0
12, 1, 2, 128, 1, 0, 0, 1
13, 3, 6, 128, 1, 0, 1, 2
```

分别表示：

1. 该进程在第12秒到达，优先级为0（实时进程），需要执行1秒，64Mb内存，无需其他I/O设备；
2. 该进程在第12秒到达，优先级为1，需要执行2秒，128Mb内存，需要1个打印机和1个光驱；
3. 该进程在第13秒到达，优先级为3，需要执行6秒，128Mb内存，需要1个打印机，1个调制解调器和2个光驱。

该文件中最多可以包含1000行，即1000个进程。

特别说明

1. 在具体的exercise07-exercise11中，我们会给出一些示例的`工作调度表`文件，它包含了一些基本的测试用例，这些测试用例将会被用作打分时的基础分。当然，最终打分时会有更多其他数据对你的调度器进行测试，所以请各位同学在自行测试时也尽可能多地考虑一些corner case，确保调度器的鲁棒性。
2. 与Project1相同，第一次tutorial后我们将会把该次tutorial所讲内容的代码给到大家，大家直接使用makefile进行编译即可得到`hostd`可执行文件。同时，最后一次tutorial过后，我们会提供一个`testp`脚本给大家用于初步测试。请确保`testp`脚本是可以正常运行的，否则助教在评阅过程中将无法评分。
3. 在每个exercise中，我们都提供了`makefile`文件，请勿修改。大家可以使用

```
make
make debug
make clean
```

来进行编译、debug、删除编译内容。

2. 项目进度

该项目的完成方式为讲练结合：教师负责对应课程内容的讲解，助教负责项目的技术指导。为了帮助同学们完成该项目，助教将组织三次习题课，逐步完善整个项目。三次习题课的内容包括：

进度	内容
一	HOST调度器简介与项目介绍。 代码框架介绍、FCFS调度算法的实现。
二	Round Robin调度算法的实现。 三级反馈调度算法的实现。
三	内存分配与资源分配。 多种调度算法的结合。

3. 项目要求

1. **使用C语言**设计一个满足上述条件的调度器。
2. 撰写一个实验报告，必须命名为`readme`，并且能够被标准文本编辑器阅读。报告包含以下内容：
    - 描述并讨论你所使用的内存分配算法，并证明你的最终设计选择；
    - 描述并讨论进程排队、调度和内存分配以及其他资源分配的结构和逻辑框架；
    - 描述并讨论你的程序的整体框架，描述代码中的各个模块和主要函数（需要对函数接口的描述）；
    - 讨论为什么HOST要使用这种多级调度策略，并与真实的操作系统方案进行对比。概述这种策略的缺点，并提出可能的改进建议。包括内存和资源分配方案。
3. 源代码必须有便于理解的注释与代码结构，便于助教评审。

4. 最终提交的文件包括源代码（只提交exercise11的`src`文件夹中的代码）、`makefile`（同样只提交exercise11的makefile）和`readme`，**不要包含可执行程序。**请将上述文件放到命名为学号的文件夹中，并压缩为zip格式，命名为`[学号].zip`并上传到Canvas作业区。例如，你的学号为520021910000，则你最终提交的压缩包中的文件树如下。

```
520021910000
    ├── makefile
    ├── readme
    └── src
        ├── hostd.c
        ├── hostd.h
        ├── mab.c
        ├── mab.h
        ├── pcb.c
        ├── pcb.h
        ├── rsrc.c
        ├── rsrc.h
        └── sigtrap.c
```

5. 特别提示：`hostd.c`的已给定框架部分、`pcb.c`、`pcb.h`以及`sigtrap.c`中包含助教的评分工具相关代码，**这部分代码请勿修改**！如果最终提交的代码因为这些原因导致无法编译通过或无法跑通，后果请自负。

4. 评分标准（满分170分）

- FCFS算法调度实时进程（10）
- 三级反馈调度（10）
- 三级反馈调度后的Round Robin模式（10）
- 上述调度的混合调度（10）
- 资源分配（10）
- 内存管理（10）
- 上述所有要求的结合（10）
- 代码结构规范、注释完整与可读性（30）
- 实验报告（70）包括如下方面：
    - 描述、讨论、证明内存分配算法的选择（20）
    - 描述并讨论调度器的结构（5）
    - 描述程序的框架和各个模块（15）
    - 讨论调度策略（包括内存和资源分配）、缺点、可能的改进方案（30）