Project 3 Interactive OS and Process Management 设计文档

中国科学院大学

葛忠鑫

2020.11.23

1. Shell 设计（非必须项）

（1） shell 实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）

在终端进行打印时应该调用 kprintf 函数，因为 kprintf 函数中使用的是 screen_write 函数打印至屏幕，才能维护光标的正常移动，以保证下一次打印的正确性。

2. kill 和 wait 内核实现的设计

（1） kill 处理过程中如何处理锁，是否有处理同步原语，如果有处理，请说明。

kill 一个进程时，会释放该进程所持有的锁，释放锁时需要将对应锁中阻塞队列里的全部任务释放。在我的设计中，kill 没有直接处理同步原语。

（2） wait 实现时，等待的进程的 PCB 用什么结构保存？

在实现 wait 时，只需将当前 PCB 移动到对应等待 PID 的 PCB 中的 wait_queue，并将状态设为 BLOCKED 即可，等待此任务退出时再释放 wait_queue 中的所有任务。

（3） 设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）

仿照 xv6 中 kill 的设计，并不在 kill 中真正杀死该进程，而是等到调度时自己 do_exit 才真正杀死自己。这样也带来一些问题，等待它的退出的进程可能不能及时地进入调度队列，选择再 do_kill 中释放其阻塞队列。

3. 同步原语设计

（1） 信号量和屏障实现的各自数据结构的包含内容

信号量：

```
1.  typedef struct semaphore
2.  {
3.      int semph;
4.      queue_t semph_queue;
5.  } semaphore_t;
```

信号量的数据结构包括一个初始信号量的值和一个信号量的阻塞队列。初始化的时候，将信号量的值初始化为初始要设的信号量的值，并初始化信号量阻塞队列。在 down 操作时，如果信号量 semph 大于 0，则将信号量减 1，否则调用 do_block 将当前进程加入阻塞队列；在 up 操作时，如果 semph_queue 队列不为空，说明有任务阻塞在该信号量上，则释放一个任务进入就绪队列，不修改 semph；否则，semph 加 1。

屏障：

```
1.  typedef struct barrier
```

```
2.  {
3.      uint32_t barry_n;
4.      uint32_t waiting_num;
5.      queue_t barry_queue;
6.  } barrier_t;
```

屏障的数据结构中 barry_n 表示要屏障的任务个数，waiting_num 是当前以及阻塞的任务个数，barry_queue 是屏障的阻塞队列；初始化的时候，将 barry_n 初始化为指定值，waiting_num 初始化为 0，并初始化 barry_queue。当有任务到达屏障的时候，对 waiting_num 加 1，并和 barry_n 比较，如果达到了屏障任务个数，则释放全部阻塞任务，然后调度；否则，调用 do_block 将当前任务添加到阻塞队列。

4. mailbox 设计

（1） mailbox 的数据结构以及主要成员变量的含义

```
1.  typedef struct mailbox
2.  {
3.      char name[32];
4.      char msg[MSG_MAX_SIZE];
5.      int msg_head, msg_tail;
6.      int used_size;
7.      int cited;
8.      condition_t full;
9.      condition_t empty;
10.     mutex_lock_t mutex;
11. } mailbox_t;
```

信箱的数据结构包括一个信息名字符串、信箱内容字符串、信箱内容的头尾指针、已使用的大小、被引用的次数、两个和空、满相关的条件变量，以及一把控制互斥访问的互斥锁。

（2） 你在 mailbox 设计中如何处理 producer-consumer 问题，使用哪种同步原语进行并发访问保护？你的实现是否支持多 producer 或多 consumer，如果有，你是如何处理的

实现中将条件变量作为同步原语的方式。生产者为写者，当信箱空间不足时，其会在 empty 条件变量上等待，直到有足够的写空间。当每次写完毕时，生产者会唤醒阻塞在 full 条件变量上的消费者（即读者）。消费者在信箱可读内容不足时，会在 full 条件变量上等待，直到有足够的读内容。当读完毕时，消费者会唤醒阻塞在 empty 条件变量上的生产者。

设计支持多个 producer 和多个 consumer。通过锁的实现，可以保证临界区只有一个进程进行访问。同时唤醒进程时，该设计采用的是 broadcast 形式，即唤醒所有相应条件变量的阻塞进程。

5. 双核使用设计（没有做双核的同学不用写）

（1）你在启用双核时遇到的问题有什么，如何解决的？

（2）你是如何让不同的任务在不同的核上运行的？

（3）你在双核上如何保证同步原语和 kill 的正确性？（做 C-Core 的同学回答）

（4）设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需

项）

尚未实现。

6. 关键函数功能

请列出上述各项功能设计里，你觉得关键的函数或代码块，及其作用