共享内存

进程间通信

- 共享内存允许两个或者多个进程共享物理内存的同一块区域（通常被称为段）。由于一个共享内存段会称为一个进程用户空间的一部分，因此这种 IPC 机制无需内核介入。所有需要做的就是让一个进程将数据复制进共享内存中，并且这部分数据会对其他所有共享同一个段的进程可用。
- 与管道等要求发送进程将数据从用户空间的缓冲区复制进内核内存和接收进程将数据从内核内存复制进用户空间的缓冲区的做法相比，这种 IPC 技术的速度更快。

- 调用 `shmget()` 创建一个新共享内存段或取得一个既有共享内存段的标识符（即由其他进程创建的共享内存段）。这个调用将返回后续调用中需要用到的共享内存标识符。
- 使用 `shmat()` 来附上共享内存段，即使该段成为调用进程的虚拟内存的一部分。
- 此刻在程序中可以像对待其他可用内存那样对待这个共享内存段。为引用这块共享内存，程序需要使用由 `shmat()` 调用返回的 `addr` 值，它是一个指向进程的虚拟地址空间中该共享内存段的起点的指针。
- 调用 `shmdt()` 来分离共享内存段。在这个调用之后，进程就无法再引用这块共享内存了。这一步是可选的，并且在进程终止时会自动完成这一步。
- 调用 `shmctl()` 来删除共享内存段。只有当当前所有附加内存段的进程都与之分离之后内存段才会销毁。只有一个进程需要执行这一步。

- `int shmget(key_t key, size_t size, int shmflg);`
- `void *shmat(int shmid, const void *shmaddr, int shmflg);`
- `int shmdt(const void *shmaddr);`
- `int shmctl(int shmid, int cmd, struct shmid_ds *buf);`
- `key_t ftok(const char *pathname, int proj_id);`

- ipcs 用法
  - ipcs -a　　// 打印当前系统中所有的进程间通信方式的信息
  - ipcs -m　　// 打印出使用共享内存进行进程间通信的信息
  - ipcs -q　　// 打印出使用消息队列进行进程间通信的信息
  - ipcs -s　　// 打印出使用信号进行进程间通信的信息
- ipcrm 用法
  - ipcrm -M shmkey　　// 移除用shmkey创建的共享内存段
  - ipcrm -m shmid　　// 移除用shmid标识的共享内存段
  - ipcrm -Q msgkey　　// 移除用msqkey创建的消息队列
  - ipcrm -q msqid　　// 移除用msqid标识的消息队列
  - ipcrm -S semkey　　// 移除用semkey创建的信号
  - ipcrm -s semid　　// 移除用semid标识的信号

牛客大学
-专业求职辅导-

THANKS

关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料