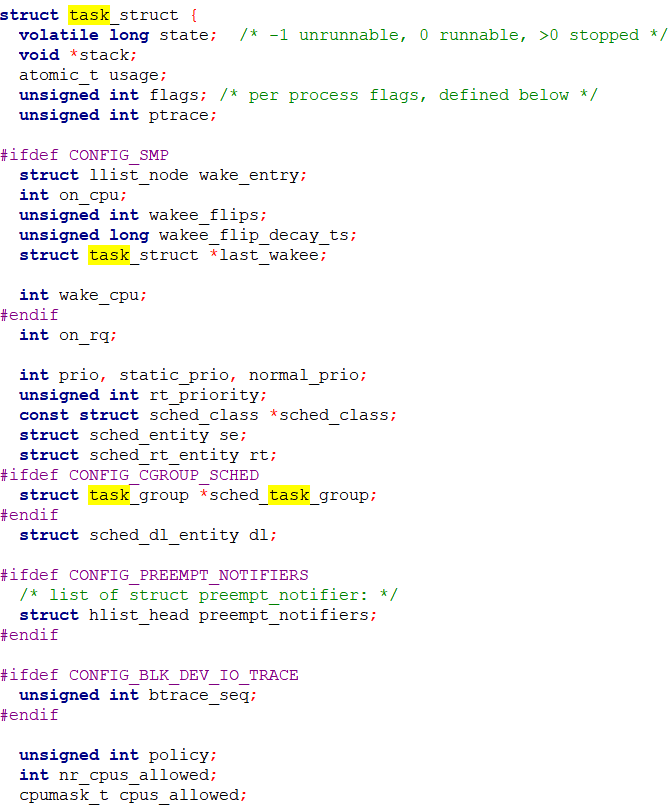
Assignment 01 实验报告

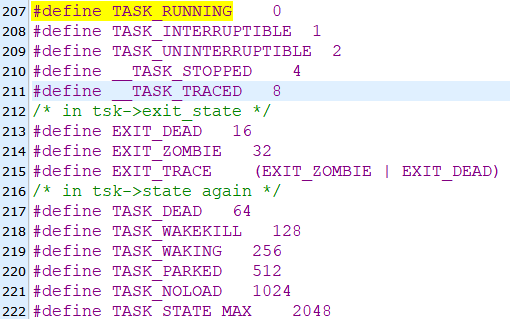
1. 哲学家就餐问题
2. 生产者-消费者问题
3. Linux内核实验
4. 内核代码fair.c阅读与CFS调度算法理解

Linux进程的基本结构、状态设置，CPU的调度基本架构，理解CFS调度算法的基本流程和主要数据结构。摘取关键代码片段，用自己的的方式描述出来。

进程基本结构：



进程状态有：



1) TASK\_RUNNING: 可运行

处于这种状态的进程，只有两种状态:

1.1) 正在运行

正在运行的进程就是当前进程(由current所指向的进程)

1.2) 正准备运行

准备运行的进程只要得到CPU就可以立即投入运行，CPU是这些进程唯一等待的系统资源，系统中有一个运行队列(run\_queue)，用来容纳所有处于可运行状态的进程，调度程序执行时，从中选择一个进程投入运行

1. TASK\_INTERRUPTIBLE: 可中断的等待状态

它是针对等待某事件或其他资源的睡眠进程设置的，在内核发送信号给该进程表明事件已经发生时，进程状态变为TASK\_RUNNING，它只要调度器选中该进程即可恢复执行

1. TASK\_UNINTERRUPTIBLE: 不可中断的等待状态

处于该状态的进程正在等待某个事件(event)或某个资源，它肯定位于系统中的某个等待队列(wait\_queue)中，处于不可中断等待态的进程是因为硬件环境不能满足而等待，例如等待特定的系统资源，它任何情况下都不能被打断，只能用特定的方式来唤醒它，例如唤醒函数wake\_up()等，它们不能由外部信号唤醒，只能由内核亲自唤醒

4) TASK\_ZOMBIE: 僵死

进程虽然已经终止，但由于某种原因，父进程还没有执行wait()系统调用，终止进程的信息也还没有回收。顾名思义，处于该状态的进程就是死进程，这种进程实际上是系统中的垃圾，必须进行相应处理以释放其占用的资源。

5) TASK\_STOPPED: 暂停

此时的进程暂时停止运行来接受某种特殊处理。通常当进程接收到SIGSTOP、SIGTSTP、SIGTTIN或 SIGTTOU信号后就处于这种状态。例如，正接受调试的进程就处于这种状态

6) TASK\_TRACED

　　从本质上来说，这属于TASK\_STOPPED状态，用于从停止的进程中，将当前被调试的进程与常规的进程区分开来

7) TASK\_DEAD

　　父进程wait系统调用发出后，当子进程退出时，父进程负责回收子进程的全部资源，子进程进入TASK\_DEAD状态

1. TASK\_SWAPPING: 换入/换出
2. 编译内核