

1. 哲学家就餐问题

osc@ubuntu: ~/final-src-osc10e/ch7

```
Philosopher4 is thinking.
Philosopher3 is thinking.
Philosopher2 is thinking.
Philosopher1 is thinking.
Philosopher0 is thinking.
Philosopher4 is hungry.
Philosopher4 is eating.
Philosopher3 is hungry.
Philosopher3 is waiting.
Philosopher2 is hungry.
Philosopher2 is eating.
Philosopher0 is hungry.
Philosopher0 is waiting.
Philosopher1 is hungry.
Philosopher1 is waiting.
Philosopher4 finished eating.
Philosopher4 is thinking.
Philosopher3 is waiting.
Philosopher0 is eating.
Philosopher2 finished eating.
Philosopher2 is thinking.
Philosopher3 is eating.
Philosopher1 is waiting.
Philosopher0 finished eating.
Philosopher1 is eating.
Philosopher0 is thinking.
Philosopher3 finished eating.
Philosopher3 is thinking.
Philosopher4 is hungry.
Philosopher4 is eating.
Philosopher2 is hungry.
Philosopher2 is waiting.
Philosopher0 is hungry.
Philosopher0 is waiting.
Philosopher3 is hungry.
Philosopher3 is waiting.
Philosopher1 finished eating.
Philosopher1 is thinking.
Philosopher0 is waiting.
Philosopher2 is eating.
Philosopher4 finished eating.
Philosopher4 is thinking.
Philosopher3 is waiting.
Philosopher0 is eating.
```

```
Philosopher1 is hungry.
Philosopher1 is waiting.
Philosopher4 is hungry.
Philosopher4 is waiting.
Philosopher2 finished eating.
Philosopher2 is thinking.
Philosopher3 is eating.
Philosopher0 finished eating.
Philosopher0 is thinking.
Philosopher4 is waiting.
Philosopher1 is eating.
Philosopher3 finished eating.
Philosopher3 is thinking.
Philosopher1 finished eating.
Philosopher1 is thinking.
Philosopher0 is hungry.
Philosopher0 is waiting.
Philosopher4 is eating.
^C
```

2. 生产者消费者问题

(1) $\lambda_p = 4$, $\lambda_c = 4$

 osc@ubuntu: ~/final-src-osc10e/ch3

```
osc@ubuntu:~/final-src-osc10e/ch3$ ./prod.o
4
pid tid and data are 1862 140579154392832 1804289383
pid tid and data are 1862 140579162785536 1681692777
pid tid and data are 1862 140579171178240 1957747793
pid tid and data are 1862 140579162785536 424238335
pid tid and data are 1862 140579171178240 596516649
pid tid and data are 1862 140579154392832 1025202362
pid tid and data are 1862 140579171178240 783368690
pid tid and data are 1862 140579154392832 2044897763
pid tid and data are 1862 140579154392832 1365180540
pid tid and data are 1862 140579162785536 304089172
pid tid and data are 1862 140579154392832 35005211
pid tid and data are 1862 140579162785536 294702567
pid tid and data are 1862 140579162785536 336465782
pid tid and data are 1862 140579171178240 278722862
pid tid and data are 1862 140579162785536 2145174067
pid tid and data are 1862 140579154392832 1101513929
pid tid and data are 1862 140579154392832 1315634022
pid tid and data are 1862 140579171178240 1369133069
pid tid and data are 1862 140579154392832 1059961393
pid tid and data are 1862 140579154392832 628175011
pid tid and data are 1862 140579162785536 1131176229
pid tid and data are 1862 140579154392832 859484421
pid tid and data are 1862 140579154392832 608413784
pid tid and data are 1862 140579162785536 1734575198
pid tid and data are 1862 140579162785536 149798315
pid tid and data are 1862 140579162785536 1129566413
pid tid and data are 1862 140579171178240 412776091
pid tid and data are 1862 140579171178240 1911759956
pid tid and data are 1862 140579154392832 137806862
pid tid and data are 1862 140579171178240 982906996
pid tid and data are 1862 140579162785536 511702305
pid tid and data are 1862 140579162785536 1937477084
pid tid and data are 1862 140579162785536 572660336
pid tid and data are 1862 140579162785536 805750846
pid tid and data are 1862 140579162785536 1100661313
pid tid and data are 1862 140579162785536 1141616124
pid tid and data are 1862 140579171178240 939819582
pid tid and data are 1862 140579171178240 1998898814
pid tid and data are 1862 140579171178240 610515434
pid tid and data are 1862 140579171178240 1374344043
pid tid and data are 1862 140579154392832 1477171087
pid tid and data are 1862 140579171178240 945117276
```


 osc@ubuntu: ~/final-src-osc10e/ch3

osc@ubuntu:~/final-src-osc10e/ch3\$./cons.o

4

```
pid tid and data are 1858 139904198747904 1804289383
pid tid and data are 1858 139904207140608 1681692777
pid tid and data are 1858 139904215533312 1590079444
pid tid and data are 1858 139904198747904 424238335
pid tid and data are 1858 139904215533312 596516649
pid tid and data are 1858 139904198747904 1025202362
pid tid and data are 1858 139904215533312 783368690
pid tid and data are 1858 139904207140608 2044897763
pid tid and data are 1858 139904207140608 304089172
pid tid and data are 1858 139904215533312 278722862
pid tid and data are 1858 139904198747904 2145174067
pid tid and data are 1858 139904215533312 336465782
pid tid and data are 1858 139904207140608 294702567
pid tid and data are 1858 139904215533312 1315634022
pid tid and data are 1858 139904198747904 1101513929
pid tid and data are 1858 139904198747904 35005211
pid tid and data are 1858 139904198747904 1365180540
pid tid and data are 1858 139904198747904 1369133069
pid tid and data are 1858 139904215533312 1059961393
pid tid and data are 1858 139904207140608 628175011
pid tid and data are 1858 139904198747904 1131176229
pid tid and data are 1858 139904207140608 859484421
pid tid and data are 1858 139904207140608 137806862
pid tid and data are 1858 139904215533312 982906996
pid tid and data are 1858 139904215533312 1911759956
pid tid and data are 1858 139904207140608 805750846
pid tid and data are 1858 139904215533312 1100661313
pid tid and data are 1858 139904207140608 1141616124
pid tid and data are 1858 139904198747904 572660336
pid tid and data are 1858 139904198747904 1937477084
pid tid and data are 1858 139904215533312 1477171087
pid tid and data are 1858 139904198747904 1374344043
pid tid and data are 1858 139904198747904 610515434
pid tid and data are 1858 139904198747904 1998898814
pid tid and data are 1858 139904215533312 1780695788
pid tid and data are 1858 139904207140608 945117276
pid tid and data are 1858 139904215533312 939819582
pid tid and data are 1858 139904207140608 511702305
pid tid and data are 1858 139904207140608 412776091
pid tid and data are 1858 139904215533312 752392754
pid tid and data are 1858 139904198747904 945117276
```


(2) $\lambda_p = 1$, $\lambda_c = 4$

 osc@ubuntu: ~/final-src-osc10e/ch3

```
osc@ubuntu:~/final-src-osc10e/ch3$ ./prod.o
1
pid tid and data are 1889 139717375125248 1804289383
pid tid and data are 1889 139717383517952 1681692777
pid tid and data are 1889 139717375125248 1714636915
pid tid and data are 1889 139717391910656 719885386
pid tid and data are 1889 139717375125248 1649760492
pid tid and data are 1889 139717391910656 1025202362
pid tid and data are 1889 139717391910656 783368690
pid tid and data are 1889 139717391910656 2044897763
pid tid and data are 1889 139717391910656 1365180540
pid tid and data are 1889 139717391910656 304089172
pid tid and data are 1889 139717391910656 35005211
pid tid and data are 1889 139717383517952 294702567
pid tid and data are 1889 139717383517952 336465782
pid tid and data are 1889 139717383517952 278722862
pid tid and data are 1889 139717375125248 2145174067
pid tid and data are 1889 139717391910656 1101513929
pid tid and data are 1889 139717391910656 1315634022
pid tid and data are 1889 139717375125248 1369133069
pid tid and data are 1889 139717375125248 1059961393
pid tid and data are 1889 139717375125248 628175011
pid tid and data are 1889 139717375125248 1131176229
pid tid and data are 1889 139717375125248 859484421
pid tid and data are 1889 139717375125248 608413784
pid tid and data are 1889 139717391910656 1734575198
pid tid and data are 1889 139717391910656 149798315
pid tid and data are 1889 139717391910656 1129566413
pid tid and data are 1889 139717383517952 412776091
pid tid and data are 1889 139717383517952 1911759956
pid tid and data are 1889 139717375125248 137806862
pid tid and data are 1889 139717383517952 982906996
pid tid and data are 1889 139717391910656 511702305
pid tid and data are 1889 139717391910656 1937477084
pid tid and data are 1889 139717383517952 572660336
pid tid and data are 1889 139717391910656 805750846
pid tid and data are 1889 139717375125248 1100661313
pid tid and data are 1889 139717391910656 1141616124
pid tid and data are 1889 139717383517952 939819582
pid tid and data are 1889 139717375125248 1998898814
pid tid and data are 1889 139717383517952 610515434
pid tid and data are 1889 139717383517952 1374344043
pid tid and data are 1889 139717375125248 1477171087
pid tid and data are 1889 139717391910656 945117276
```


 osc@ubuntu: ~/final-src-osc10e/ch3

```
osc@ubuntu:~/final-src-osc10e/ch3$ ./cons.o
```

```
4
```

```
pid tid and data are 1885 140148169844480 1804289383
pid tid and data are 1885 140148178237184 1365180540
pid tid and data are 1885 140148186629888 1714636915
pid tid and data are 1885 140148169844480 1365180540
pid tid and data are 1885 140148186629888 1649760492
pid tid and data are 1885 140148169844480 1025202362
pid tid and data are 1885 140148186629888 783368690
pid tid and data are 1885 140148178237184 1129566413
pid tid and data are 1885 140148178237184 1911759956
pid tid and data are 1885 140148186629888 137806862
pid tid and data are 1885 140148169844480 982906996
pid tid and data are 1885 140148186629888 511702305
pid tid and data are 1885 140148186629888 1937477084
pid tid and data are 1885 140148169844480 572660336
pid tid and data are 1885 140148178237184 805750846
pid tid and data are 1885 140148178237184 1100661313
pid tid and data are 1885 140148178237184 412776091
pid tid and data are 1885 140148178237184 939819582
pid tid and data are 1885 140148169844480 939819582
pid tid and data are 1885 140148178237184 610515434
pid tid and data are 1885 140148186629888 1374344043
pid tid and data are 1885 140148178237184 1477171087
pid tid and data are 1885 140148169844480 945117276
pid tid and data are 1885 140148178237184 1780695788
pid tid and data are 1885 140148178237184 491705403
pid tid and data are 1885 140148169844480 752392754
pid tid and data are 1885 140148178237184 2053999932
pid tid and data are 1885 140148169844480 1411549676
pid tid and data are 1885 140148186629888 943947739
pid tid and data are 1885 140148186629888 855636226
pid tid and data are 1885 140148178237184 1469348094
pid tid and data are 1885 140148186629888 1036140795
pid tid and data are 1885 140148186629888 2040651434
pid tid and data are 1885 140148178237184 317097467
pid tid and data are 1885 140148186629888 1376710097
pid tid and data are 1885 140148169844480 1330573317
pid tid and data are 1885 140148186629888 1687926652
pid tid and data are 1885 140148169844480 959997301
pid tid and data are 1885 140148169844480 402724286
pid tid and data are 1885 140148186629888 1194953865
pid tid and data are 1885 140148178237184 364228444
```


(3) $\lambda_p = 4$, $\lambda_c = 1$

 osc@ubuntu: ~/final-src-osc10e/ch3

```
osc@ubuntu:~/final-src-osc10e/ch3$ ./prod.o
4
pid tid and data are 2235 139954204157696 1804289383
pid tid and data are 2235 139954212550400 1681692777
pid tid and data are 2235 139954220943104 1957747793
pid tid and data are 2235 139954212550400 719885386
pid tid and data are 2235 139954212550400 596516649
pid tid and data are 2235 139954204157696 1025202362
pid tid and data are 2235 139954212550400 783368690
pid tid and data are 2235 139954204157696 2044897763
pid tid and data are 2235 139954204157696 1365180540
pid tid and data are 2235 139954204157696 304089172
pid tid and data are 2235 139954212550400 35005211
pid tid and data are 2235 139954220943104 294702567
pid tid and data are 2235 139954220943104 336465782
pid tid and data are 2235 139954204157696 278722862
pid tid and data are 2235 139954220943104 2145174067
pid tid and data are 2235 139954212550400 1101513929
pid tid and data are 2235 139954212550400 1315634022
pid tid and data are 2235 139954204157696 1369133069
pid tid and data are 2235 139954212550400 1059961393
pid tid and data are 2235 139954212550400 628175011
pid tid and data are 2235 139954220943104 1131176229
pid tid and data are 2235 139954212550400 859484421
pid tid and data are 2235 139954212550400 608413784
pid tid and data are 2235 139954220943104 1734575198
pid tid and data are 2235 139954220943104 149798315
pid tid and data are 2235 139954220943104 1129566413
pid tid and data are 2235 139954204157696 412776091
pid tid and data are 2235 139954204157696 1911759956
pid tid and data are 2235 139954212550400 137806862
pid tid and data are 2235 139954204157696 982906996
pid tid and data are 2235 139954220943104 511702305
pid tid and data are 2235 139954220943104 1937477084
pid tid and data are 2235 139954220943104 572660336
pid tid and data are 2235 139954220943104 805750846
pid tid and data are 2235 139954220943104 1100661313
pid tid and data are 2235 139954220943104 1141616124
pid tid and data are 2235 139954204157696 939819582
pid tid and data are 2235 139954204157696 1998898814
pid tid and data are 2235 139954204157696 610515434
pid tid and data are 2235 139954212550400 1374344043
pid tid and data are 2235 139954204157696 1477171087
```

osc@ubuntu: ~/final-src-osc10e/ch3

osc@ubuntu:~/final-src-osc10e/ch3\$./cons.o

1

```
pid tid and data are 2239 139956180100864 783368690
pid tid and data are 2239 139956188493568 1025202362
pid tid and data are 2239 139956180100864 596516649
pid tid and data are 2239 139956188493568 719885386
pid tid and data are 2239 139956180100864 1957747793
pid tid and data are 2239 139956188493568 1681692777
pid tid and data are 2239 139956180100864 1804289383
pid tid and data are 2239 139956196886272 1365180540
pid tid and data are 2239 139956196886272 2044897763
pid tid and data are 2239 139956180100864 35005211
pid tid and data are 2239 139956188493568 304089172
pid tid and data are 2239 139956180100864 336465782
pid tid and data are 2239 139956188493568 294702567
pid tid and data are 2239 139956196886272 278722862
pid tid and data are 2239 139956180100864 2145174067
pid tid and data are 2239 139956188493568 1315634022
pid tid and data are 2239 139956196886272 1101513929
pid tid and data are 2239 139956188493568 628175011
pid tid and data are 2239 139956180100864 1059961393
pid tid and data are 2239 139956196886272 1369133069
pid tid and data are 2239 139956188493568 1131176229
pid tid and data are 2239 139956196886272 859484421
pid tid and data are 2239 139956180100864 608413784
pid tid and data are 2239 139956196886272 412776091
pid tid and data are 2239 139956196886272 1129566413
pid tid and data are 2239 139956180100864 149798315
pid tid and data are 2239 139956180100864 1734575198
pid tid and data are 2239 139956196886272 1911759956
pid tid and data are 2239 139956180100864 137806862
pid tid and data are 2239 139956188493568 982906996
pid tid and data are 2239 139956180100864 511702305
pid tid and data are 2239 139956196886272 1937477084
pid tid and data are 2239 139956188493568 572660336
pid tid and data are 2239 139956180100864 805750846
pid tid and data are 2239 139956196886272 1100661313
pid tid and data are 2239 139956188493568 1141616124
pid tid and data are 2239 139956180100864 939819582
pid tid and data are 2239 139956196886272 1998898814
pid tid and data are 2239 139956188493568 610515434
pid tid and data are 2239 139956180100864 1374344043
pid tid and data are 2239 139956196886272 1477171087
```


3. Linux 内核实验

a.

Linux 进程的基本结构、状态设置

进程由可执行的指令代码，数据和堆栈区组成。Linux 操作系统在内核中使用 `task_struct` 结构体来记录一个进程的各种信息，这个结构体的实例就是进程描述符，主要包括进程当前运行的状态信息、内核栈信息、进程使用状态、PID、优先级、锁、时间片、队列、信号量、内存管理信息、文件列表等等与进程管理、调度密切相关的信息。

linux-5.3.8\include\linux 目录下的 `sched.h` 文件中定义了 `task_struct` 结构体，部分代码如下：

```
637 struct task_struct {
638     #ifdef CONFIG_THREAD_INFO_IN_TASK
639         /*
640          * For reasons of header soup (see current_thread_info()), this
641          * must be the first element of task_struct.
642          */
643         struct thread_info    thread_info;
644     #endif
645     /* -1 unrunnable, 0 runnable, >0 stopped: */
646     volatile long            state;
647
648     /*
649      * This begins the randomizable portion of task_struct. Only
650      * scheduling-critical items should be added above here.
651      */
652     randomized_struct_fields_start
653
654     void                    *stack;
655     refcount_t              usage;
```

上述定义的 `state` 作为进程的状态标志，为 -1 时表示当前进程不可运行，为 0 是可运行，大于 0 时表示进程已停止。

```
683     int                    prio;
684     int                    static_prio;
685     int                    normal_prio;
686     unsigned int           rt_priority;
687
688     const struct sched_class *sched_class;
689     struct sched_entity    se;
690     struct sched_rt_entity rt;
691     #ifdef CONFIG_CGROUP_SCHED
692     struct task_group      *sched_task_group;
693     #endif
694     struct sched_dl_entity dl;
```

上述过程中定义了优先级，`prio` 和 `normal_prio` 表示动态优先级，`static_prio` 表示进程的静态优先级。静态优先级是进程启动时分配的优先级，它可以用 `nice` 和 `sched_setscheduler` 系统调用修改，否则在进程运行期间会一直保持恒定。`normal_prio` 表示基于进程的静态优先级和调度策略计算出的优先级。调度器考虑的优先级则保存在 `prio`。由于在某些情况下内核需要暂时提高进程的优先级，因此需要第 3 个成员来表示。


```

749     int          exit_state;
750     int          exit_code;
751     int          exit_signal;
752     /* The signal sent when the parent dies: */
753     int          pdeath_signal;

```

定义状态、代码、信号相关信息，753 行定义一个父进程“死”后的信号。

```

801     pid_t        pid;
802     pid_t        tgid;

```

定义进程的 pid，进程的组号 tgid。

```

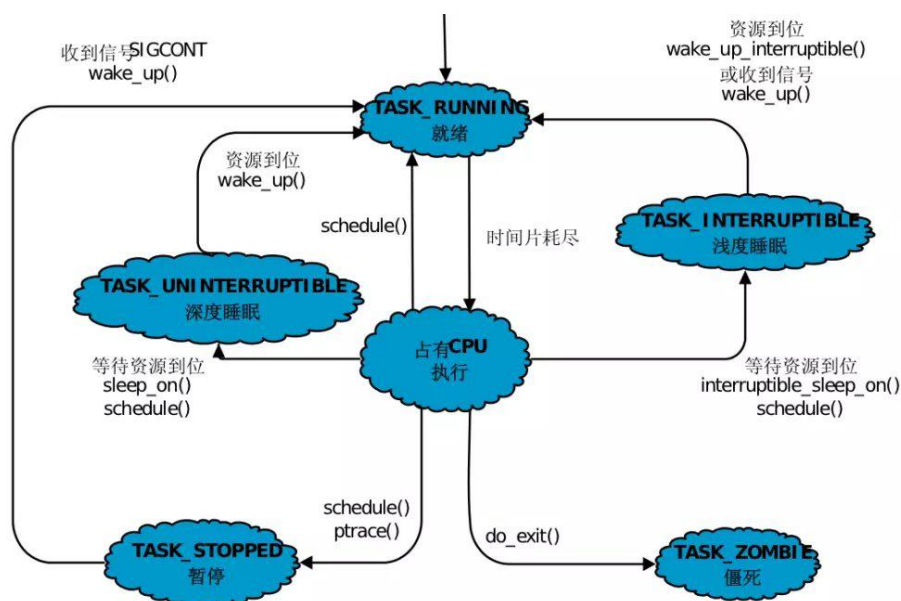
808     /*
809     * Pointers to the (original) parent process, youngest child, younger sibling,
810     * older sibling, respectively. (p->father can be replaced with
811     * p->real_parent->pid)
812     */
813
814     /* Real parent process: */
815     struct task_struct __rcu *real_parent;
816
817     /* Recipient of SIGCHLD, wait4() reports: */
818     struct task_struct __rcu *parent;
819
820     /*
821     * Children/sibling form the list of natural children:
822     */
823     struct list_head children;
824     struct list_head sibling;
825     struct task_struct *group_leader;

```

定义指针指向真实的父进程, SIGCHLD 的接收者（现在的父进程），并且定义了子进程、兄弟进程列表。825 行定义了线程组的 leader。

Linux 中，进程的组织结构是双向链表的形式，task 中有 prev 和 next 的指针，其中链表表头 head 的位置是 0 号进程或者说是 swapper 进程的 task_struct。SET_LINKS 和 REMOVE_LINKS 宏分别用于从进程链表中插入和删除一个进程描述符。这些宏考虑了进程间的父子关系。

进程状态转换图：



进程可分为 5 种状态：运行状态、可中断睡眠状态、不可中断睡眠状态、暂停状态、僵死状态。各自的含义如下：当进程正在被 CPU 执行，或已经准备就绪随时可由调度程序执行，则称该进程为处于运行状态。可中断睡眠状态为当进程处于可中断等待状态时，系统不会调度该进程执行，当系统产生一个中断或者释放了进程正在等待的资源，或者进程收到一个信号，都可以唤醒进程转换到就绪状态（运行状态）。不可中断睡眠状态与可中断睡眠状态类似，但处于该状态的进程只有被使用 `wake_up()` 函数明确唤醒时才能转换到可运行的就绪状态。当进程收到信号 `SIGSTOP`、`SIGTSTP`、`SIGTTIN` 或 `SIGTTOU` 时就会进入暂停状态，向其发送 `SIGCONT` 信号让进程转换到可运行状态。当进程已停止运行，但其父进程还没有询问其状态时，则称该进程处于僵死状态[1]。

CPU 的调度基本架构

CPU 调度的任务是，从就绪队列中选择一个等待进程，并为其分配 CPU。调度程序分配 CPU 到选中的进程。在目前 Linux 内核中，调度器分成两个层级，在进程中被直接调用的成为通用调度器或者核心调度器。

当主调度器和周期调度器进行进程调度时先查询调度器类，再根据优先级选择进程。这里涉及调度策略、调度类和调度实体。调度类在管理和选择进程时不会直接操作进程，而是通过调度实体进行操作，因为调度器可以操作比进程更一般的实体，所以需要有一个数据结构来描述此类实体就是调度实体，调度实体中包含很多关于进程运行调度信息，比如 `sched_entity` 中 `load` 负载均衡的权重，决定实体占队列总负荷比例，CFS 在计算虚拟时间的速度会依赖总负荷；`run_node` 标准树节点，在红黑树上的排序；`on_rq` 实体是否在就绪队列上；`sum_exec_runtime` 在进程运行时，CFS 中记录进程消耗 CPU 时间，也就是每次新进程加入就绪队列或周期性调度器时，当前时间和 `exec_start` 差值累加到 `sum_exec_runtime` 中，而 `exec_start` 则更新为当前时间；`prev_sum_exec_runtime` 在进程调度不再占用 CPU 时，`sum_exec_runtime` 保存到 `prev_sum_exec_runtime` 中；这些信息都是进程调度所必须的，所以每个 `task_struct` 中都有一个实体，进程都是可调度实体，但是可调度实体不一定是进程[3]。

调度策略	调度实体	调度类
SCHED_NORMAL、SCHED_BATCH	<code>sched_entity</code>	<code>fair_sched_class</code>
SCHED_FIFO、SCHED_RR	<code>sched_rt_entity</code>	<code>rt_sched_class</code>
SCHED_IDLE	—	<code>idle_sched_class</code>

调度策略定义在 `linux-5.3.8/include/uapi/linux/sched.h` 中，代码如下：

fair.c

sched.h

sched.h

```
52  /*
53   * Scheduling policies
54   */
55  #define SCHED_NORMAL      0
56  #define SCHED_FIFO        1
57  #define SCHED_RR          2
58  #define SCHED_BATCH       3
59  /* SCHED_ISO: reserved but not implemented yet */
60  #define SCHED_IDLE        5
61  #define SCHED_DEADLINE    6
```

SCHED_NORMAL：普通的分时进程，使用的 `fair_sched_class` 调度类。**SCHED_FIFO**：先进先出的实时进程。当调用程序把 CPU 分配给进程的时候，它把该进程描述符保留在运行队列链表的当前位置。此调度策略的进程一旦使用 CPU 则一直运行。如果没有其他可运行

的更高优先级实时进程，进程就继续使用 CPU，想用多久就用多久，即使还有其他具有相同优先级的实时进程处于可运行状态。使用的是 `rt_sched_class` 调度类。**SCHED_RR**：时间片轮转的实时进程。当调度程序把 CPU 分配给进程的时候，它把该进程的描述符放在运行队列链表的末尾。这种策略保证对所有具有相同优先级的 **SCHED_RR** 实时进程进行公平分配 CPU 时间，使用的 `rt_sched_class` 调度类。**SCHED_BATCH**：是 **SCHED_NORMAL** 的分化版本。采用分时策略，根据动态优先级，分配 CPU 资源。在有实时进程的时候，实时进程优先调度。但针对吞吐量优化，除了不能抢占外与常规进程一样，允许任务运行更长时间，更好使用高速缓存，适合于成批处理的工作，使用的 `fair_sched_class` 调度类。**SCHED_IDLE**：优先级最低，在系统空闲时运行，使用的是 `idle_sched_class` 调度类，给 0 号进程使用。**SCHED_DEADLINE**：新支持的实时进程调度策略，针对突发型计算，并且对延迟和完成时间敏感的任务使用，基于 EDF（earliest deadline first），使用的是 `dl_sched_class` 调度类[2]。

接下来的调度类和调度实体均用 CFS 调度算法的相关代码来说明。

`sched_class` 是 Linux 内核为不同调度策略定义的调度类（定义在 `linux-5.3.8\kernel\sched\sched.h` 文件中），`fair_sched_class` 是以 `sched_class` 结构为基础，定义在 `linux-5.3.8\kernel\sched\fair.c` 中。

```

10444  /*
10445   * All the scheduling class methods:
10446   */
10447  const struct sched_class fair_sched_class = {
10448      .next                = &idle_sched_class,
10449      .enqueue_task        = enqueue_task_fair,
10450      .dequeue_task        = dequeue_task_fair,
10451      .yield_task          = yield_task_fair,
10452      .yield_to_task       = yield_to_task_fair,
10453
10454      .check_preempt_curr  = check_preempt_wakeup,
10455
10456      .pick_next_task      = pick_next_task_fair,
10457      .put_prev_task       = put_prev_task_fair,
10458
10459      #ifdef CONFIG_SMP
10460      .select_task_rq      = select_task_rq_fair,
10461      .migrate_task_rq     = migrate_task_rq_fair,
10462
10463      .rq_online           = rq_online_fair,
10464      .rq_offline          = rq_offline_fair,
10465
10466      .task_dead           = task_dead_fair,
10467      .set_cpus_allowed    = set_cpus_allowed_common,
10468      #endif
10469
10470      .set_curr_task       = set_curr_task_fair,
10471      .task_tick           = task_tick_fair,
10472      .task_fork           = task_fork_fair,
10473
10474      .prio_changed        = prio_changed_fair,
10475      .switched_from       = switched_from_fair,
10476      .switched_to         = switched_to_fair,
10477
10478      .get_rr_interval     = get_rr_interval_fair,
10479
10480      .update_curr         = update_curr_fair,
10481

```

```

10482 #ifdef CONFIG_FAIR_GROUP_SCHED
10483     .task_change_group = task_change_group_fair,
10484 #endif
10485
10486 #ifdef CONFIG_UCLAMP_TASK
10487     .uclamp_enabled = 1,
10488 #endif
10489 };
10490

```

Next:指向下一个调度类；enqueue_task:将任务加入到调度类中；dequeue_task:将任务从调度类中移除；yield_task/ yield_to_task:主动放弃 CPU；check_preempt_curr:检查当前进程是否可被强占；pick_next_task:从调度类中选出下一个要运行的进程；put_prev_task:将进程放回到调度类中；select_task_rq:为进程选择一个合适的 cpu 的运行队列；migrate_task_rq:迁移到另外的 cpu 运行队列；rq_online:启动运行队列；rq_offline:关闭运行队列；task_dead:进程结束时调用；set_cpus_allowed:修改进程 cpu 亲和力 affinity；set_curr_task:当进程改变调度类或者进程组时被调用 task_tick:将会引起进程切换，驱动运行 running 强占，由 time_tick 调用；task_fork:进程创建时调用，不同调度策略的进程初始化不一样；prio_changed:改变进程优先级；switched_from、switched_to:进程改变调度器时使用。

linux-5.3.8\include\linux\sched.h 中定义了 sched_entity，代码如下：

```

fair.c sched.h
457 struct sched_entity {
458     /* For load-balancing: */
459     struct load_weight    load;
460     unsigned long         runnable_weight;
461     struct rb_node        run_node;
462     struct list_head      group_node;
463     unsigned int          on_rq;
464
465     u64                    exec_start;
466     u64                    sum_exec_runtime;
467     u64                    vruntime;
468     u64                    prev_sum_exec_runtime;
469
470     u64                    nr_migrations;
471
472     struct sched_statistics statistics;
473
474 #ifdef CONFIG_FAIR_GROUP_SCHED
475     int                    depth;
476     struct sched_entity    *parent;
477     /* rq on which this entity is (to be) queued: */
478     struct cfs_rq          *cfs_rq;
479     /* rq "owned" by this entity/group: */
480     struct cfs_rq          *my_rq;
481 #endif
482
483 #ifdef CONFIG_SMP
484     /*
485      * Per entity load average tracking.
486      *
487      * Put into separate cache line so it does not
488      * collide with read-mostly values above.
489      */
490     struct sched_avg       avg;
491 #endif
492 };
493

```

459-460 行定义了权重，run_node 为实体在红黑树对应结点的信息，group_node 为实体所在的进程组，on_rq 标志实体是否处于红黑树运行队列中。开始运行时间为 exec_start,总运行时间为 sum_exec_runtime,vruntime 为虚拟运行时间，进程在切换进 CPU 时的 prev_sum_exec_runtime 值，调度实体中进程移到其他 CPU 组的数量为 nr_migrations。*cfs_rq

为实体所处红黑树运行队列，*my_q 为实体的红黑树运行队列，如果为 NULL 表明其是一个进程，若非 NULL 表明其是调度组。

CFS 调度算法的基本流程和主要数据结构

主要数据结构中的调度类和调度实体在前面陈述过。

CFS 调度算法为每个任务分配一定比例的 CPU 处理时间，每个任务分配的具体比例根据友好值来计算的。CFS 采用目标延迟即每个可运行任务应当运行一次的时间间隔，根据目标延迟按比例分配 CPU 时间。CFS 调度程序通过每个任务的变量 `vruntime` 以便维护虚拟运行时间，进而记录每个任务运行多久。每个变成可运行任务的任务会被放置在红黑树上，变成不可行则从树上删除。红黑树的键是基于 `vruntime` 值，得到较少处理时间的任务（`vruntime` 值较小）会偏向树的左侧，反之偏向树的右侧。根据二分搜索树的性质，最左侧的结点有最小的键值，同时也是具有最高优先级的任务。当调度器从根 CFS 运行队列中选择一个进程组进行调度时，进程组会从自己的 CFS 运行队列中选择一个调度实体进行调度（这个调度实体可能为进程，也可能又是一个子进程组），就这样一直深入，直到最后选出一个进程进行运行为止。根据 `rq->cfs_rq` 中进程的数量计算一个总的 period 周期，每个进程再根据自己的 `weight` 占整个的比重来计算自己的理想运行时间，在 `scheduler_tick()` 中判断如果进程的实际运行时间(`exec_runtime`)已经达到理想运行时间，则进程需要被调度。有了 period，那么 `cfs_rq` 中所有进程在 period 以内必会得到调度。

接下来的函数调用在 `linux-5.3.8\kernel\sched\fair.c` 中定义。

```
291 static inline bool list_add_leaf_cfs_rq(struct cfs_rq *cfs_rq)
```

该函数对新加入的任务进行关于父进程相关判断的操作（加入红黑树）。

```
359 static inline void list_del_leaf_cfs_rq(struct cfs_rq *cfs_rq)
```

该函数从红黑树中将开始运行的任务移除。

```
565 static void __enqueue_entity(struct cfs_rq *cfs_rq, struct sched_entity *se)
```

该函数将实体加入到红黑树中。

```
662 static inline u64 calc_delta_fair(u64 delta, struct sched_entity *se)
663 {
664     if (unlikely(se->load.weight != NICE_0_LOAD))
665         delta = __calc_delta(delta, NICE_0_LOAD, &se->load);
666
667     return delta;
668 }
```

该函数定义如何计算时间。

```
834 static void update_curr(struct cfs_rq *cfs_rq)
```

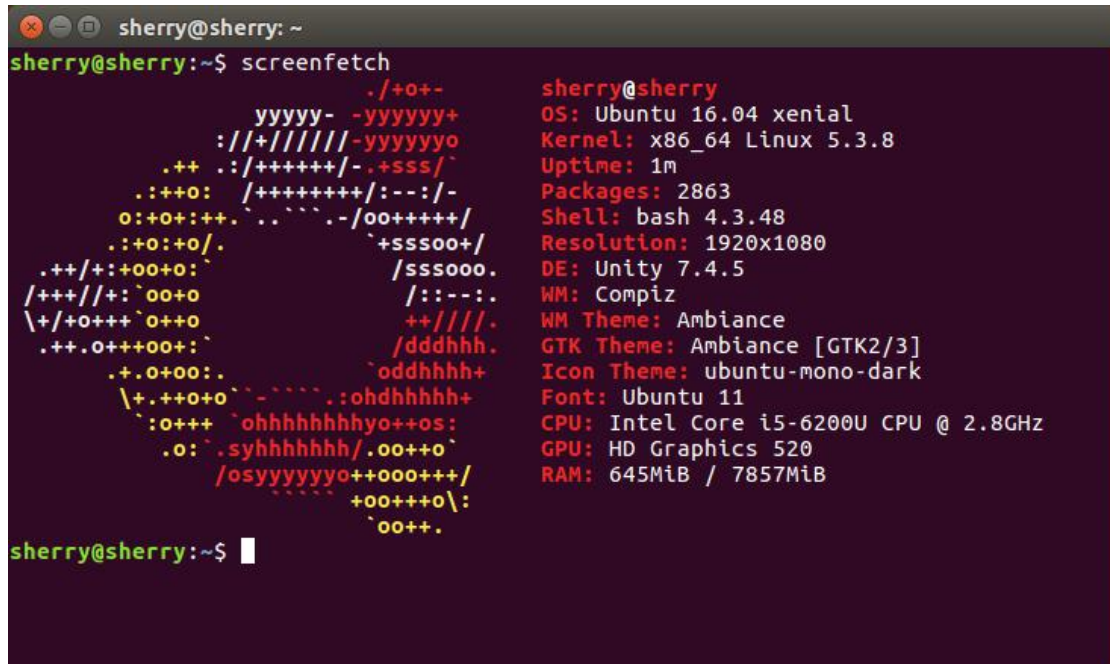
该函数不断更新当前任务的运行时间，运行时间对于红黑树的构建是很重要的。

```
992 /*
993  * Task is being enqueued - update stats:
994  */
995 static inline void
996 update_stats_enqueue(struct cfs_rq *cfs_rq, struct sched_entity *se, int flags)
```

任务入队则需更新状态。

在 `linux-5.3.8\kernel\sched\sched.h` 文件中，对 CFS 公平调度器的就绪队列 `cfs_rq`、实时进程就绪队列 `rt_rq`、deadline 就绪队列 `dl_rq` 结构进行定义。

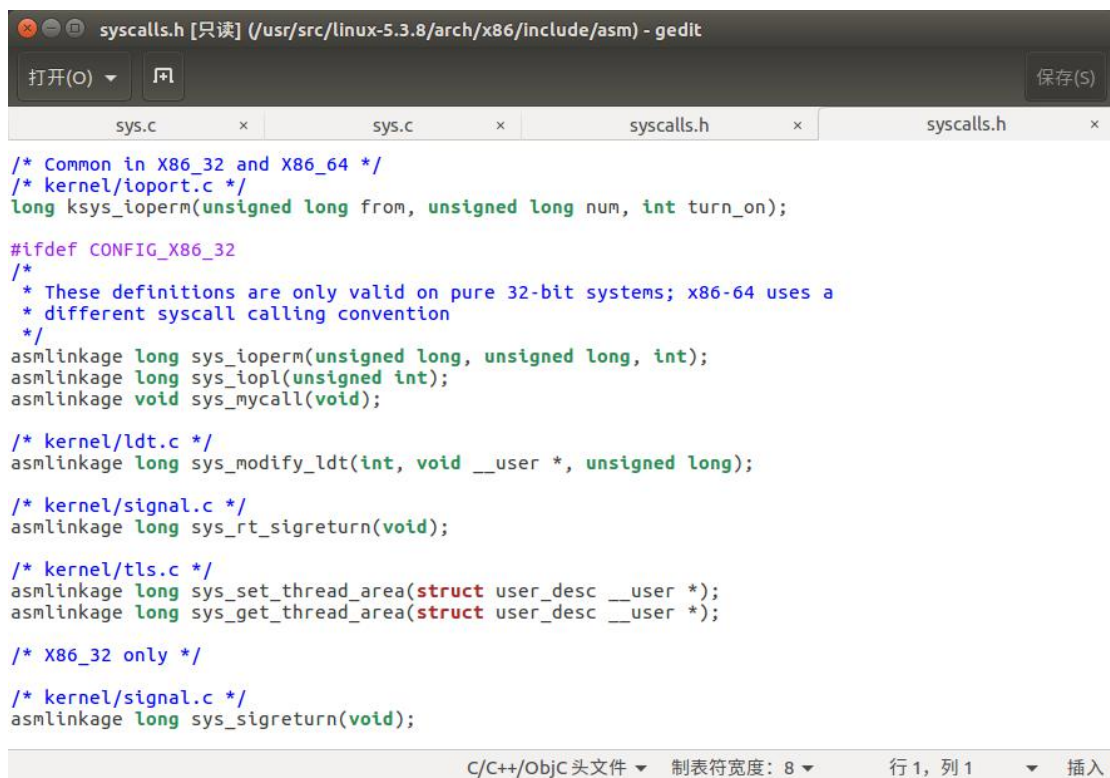
内核重新编译后的图片（中间有扩容的步骤）



```
sherry@sherry: ~  
sherry@sherry:~$ screenfetch  
      ./+o+-  
      yyyyy- -yyyyyy+  
      ://+////////-yyyyyyo  
      .++ .:/++++++/-..+sss/  
      .:++o: /+++++++/:--:/-  
      o:+o+:++ ..`.-/oo+++++/  
      .:o:+o/. `+sss00+/  
      .++/+::oo+o:` /sss00o.  
      /+++//+:`oo+o /:--:..  
      \+/+o+++`o+o ++//..  
      .++..o+++oo+:` /dddhhh.  
      .+.o+oo:.. `oddhhhh+  
      \+.++o+o`-`-`-`-:ohdhhhh+  
      :o+++`ohdhhhhhyo++os:  
      .o:`.syhhhhhhy/.oo++o`  
      /osyyyyyyo++ooo+++/  
      `++++` +oo++o\:  
      `oo++.  
sherry@sherry:~$
```

sherry@sherry
OS: Ubuntu 16.04 xenial
Kernel: x86_64 Linux 5.3.8
Uptime: 1m
Packages: 2863
Shell: bash 4.3.48
Resolution: 1920x1080
DE: Unity 7.4.5
WM: Compiz
WM Theme: Ambiance
GTK Theme: Ambiance [GTK2/3]
Icon Theme: ubuntu-mono-dark
Font: Ubuntu 11
CPU: Intel Core i5-6200U CPU @ 2.8GHz
GPU: HD Graphics 520
RAM: 645MiB / 7857MiB

修改的部分:



```
syscalls.h [只读] (/usr/src/linux-5.3.8/arch/x86/include/asm) - gedit  
打开(O) 保存(S)  
sys.c x sys.c x syscalls.h x syscalls.h x  
/* Common in X86_32 and X86_64 */  
/* kernel/ioport.c */  
long ksys_ioperm(unsigned long from, unsigned long num, int turn_on);  
  
#ifdef CONFIG_X86_32  
/*  
 * These definitions are only valid on pure 32-bit systems; x86-64 uses a  
 * different syscall calling convention  
 */  
asmlinkage long sys_ioperm(unsigned long, unsigned long, int);  
asmlinkage long sys_iopl(unsigned int);  
asmlinkage void sys_mycall(void);  
  
/* kernel/ldt.c */  
asmlinkage long sys_modify_ldt(int, void __user *, unsigned long);  
  
/* kernel/signal.c */  
asmlinkage long sys_rt_sigreturn(void);  
  
/* kernel/tls.c */  
asmlinkage long sys_set_thread_area(struct user_desc __user *);  
asmlinkage long sys_get_thread_area(struct user_desc __user *);  
  
/* X86_32 only */  
  
/* kernel/signal.c */  
asmlinkage long sys_sigreturn(void);  
C/C++/Objc 头文件 制表符宽度: 8 行 1, 列 1 插入
```

```
sys.c [只读] (/usr/src/linux-5.3.8/kernel) - gedit
保存(S)

sys.c
__put_user(s.buffer_id, &info->buffer_id) ||
__put_user(s.totalswap, &info->totalswap) ||
__put_user(s.freeswap, &info->freeswap) ||
__put_user(s.procs, &info->procs) ||
__put_user(s.totalhigh, &info->totalhigh) ||
__put_user(s.freehigh, &info->freehigh) ||
__put_user(s.mem_unit, &info->mem_unit))
return -EFAULT;

return 0;
}
asmlinkage void sys_mycall(void)
{
    int i=0;
    printk("se.exec_start:%llu\n", current->se.exec_start);
    printk("se.vruntime:%llu\n", current->se.vruntime);
    printk("se.sum_exec_runtime:%llu\n", current->se.sum_exec_runtime);
    printk("se.nr_migrations:%llu\n", current->se.nr_migrations);
    printk("nr_switches:%llu\n", current->se.cfs_rq->rq->nr_switches);
    printk("nr_voluntary_switches:%llu\n", current->se.cfs_rq->rq->nr_switches);
    printk("nr_involuntary_switches:%d\n", i);
    printk("se.load.weight:%lu\n", current->se.load.weight);
    printk("se.avg.load_sum:%llu\n", current->se.avg.load_sum);
    printk("se.avg.util_sum:%u\n", current->se.avg.util_sum);
    printk("se.avg.load_avg:%lu\n", current->se.avg.load_avg);
    printk("se.avg.util_avg:%lu\n", current->se.avg.util_avg);
    printk("se.avg.last_update_time:%llu\n", current->se.avg.last_update_time);
}
#endif /* CONFIG_COMPAT */
```

```
syscall_64.tbl [只读] (/usr/src/linux-5.3.8/arch/x86/entry/syscalls) - gedit
保存(S)

sys.c x sys.c x syscalls.h x syscalls.h x syscall_64.tbl x syscall_64.tbl x
428 common open_tree __x64_sys_open_tree
429 common move_mount __x64_sys_move_mount
430 common fsopen __x64_sys_fsopen
431 common fsconfig __x64_sys_fsconfig
432 common fsmount __x64_sys_fsmount
433 common fspick __x64_sys_fspick
434 common pidfd_open __x64_sys_pidfd_open
435 common clone3 __x64_sys_clone3/ptregs
436 64 mycall sys_mycall
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation. The __x32_compat_sys stubs are created
# on-the-fly for compat_sys_*( ) compatibility system calls if X86_X32
# is defined.
#
512 x32 rt_sigaction __x32_compat_sys_rt_sigaction
513 x32 rt_sigreturn sys32_x32_rt_sigreturn
514 x32 ioctl __x32_compat_sys_ioctl
515 x32 readv __x32_compat_sys_readv
516 x32 writev __x32_compat_sys_writev
517 x32 recvfrom __x32_compat_sys_recvfrom
518 x32 sendmsg __x32_compat_sys_sendmsg
519 x32 recvmsg __x32_compat_sys_recvmsg
520 x32 execve __x32_compat_sys_execve/ptregs
521 x32 ptrace __x32_compat_sys_ptrace
522 x32 rt_sigpending __x32_compat_sys_rt_sigpending
523 x32 rt_sigtimedwait __x32_compat_sys_rt_sigtimedwait_time64
524 x32 rt_sigqueueinfo __x32_compat_sys_rt_sigqueueinfo
525 x32 sigaltstack __x32_compat_sys_sigaltstack
```

打印出的信息：


```
sherry@sherry: ~/桌面
[ 67.826362] nr_involuntary_switches:0
[ 67.826362] se.load.weight:1048576
[ 67.826363] se.avg.load_sum:46883
[ 67.826363] se.avg.util_sum:24260791
[ 67.826363] se.avg.load_avg:1024
[ 67.826364] se.avg.util_avg:517
[ 67.826364] se.avg.last_update_time:67826258944
sherry@sherry:~/桌面$ sudo dmesg -c
sherry@sherry:~/桌面$ ./mycall.o
sherry@sherry:~/桌面$ dmesg
[ 191.725548] se.exec_start:191724477970
[ 191.725551] se.vruntime:208963147
[ 191.725552] se.sum_exec_runtime:638996
[ 191.725554] se.nr_migrations:0
[ 191.725555] nr_switches:88161
[ 191.725556] nr_voluntary_switches:88161
[ 191.725557] nr_involuntary_switches:0
[ 191.725559] se.load.weight:1048576
[ 191.725560] se.avg.load_sum:47445
[ 191.725561] se.avg.util_sum:24611328
[ 191.725562] se.avg.load_avg:1024
[ 191.725563] se.avg.util_avg:512
[ 191.725565] se.avg.last_update_time:191724477440
sherry@sherry:~/桌面$
```

- [1] <https://www.cnblogs.com/KevinGeorge/p/7866778.html>
- [2] <https://blog.csdn.net/you5522393/article/details/80702772>
- [3] https://blog.csdn.net/weixin_42092278/article/details/89187728