Lab6实验报告

班级：计算机科学与技术2班

姓名：许金曼

学号：171491220

Lab6 实现Stride Scheduling调度算法

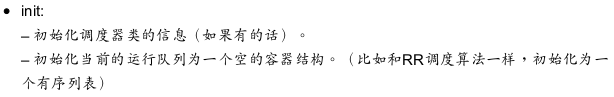
Stride Scheduling算法的基本思想：

1、为每个runnable的进程设置一个当前状态stride，表示该进程当前的调度权。另外定义其对应的pass值，表示对应进程在调度后，stride需要进行的累加值。

2、每次需要调度时，从当前runnable态的进程中选择stride最小的进程调度。

3、对于获得调度的进程，将对应的stride加上其对应的步长pass。

4、在一段固定的时间之后，回到第2步，重新调度当前stride最小的进程。



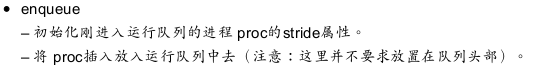
static void stride\_init(struct run\_queue \*rq){

list\_init(&(rq->run\_list));

rq->lab6\_run\_pool = NULL; //初始化当前进程运行队列为空

rq->proc\_num = 0; //设置运行队列为空

}



static void

stride\_enqueue(struct run\_queue \*rq, struct proc\_struct \*proc) {

rq->lab6\_run\_pool = skew\_heap\_insert(rq->lab6\_run\_pool,&(proc->lab6\_run\_pool), proc\_stride\_comp\_f);

#else

assert(list\_empty(&(proc->run\_link)));

list\_add\_before(&(rq->run\_list), &(proc->run\_link));

#endif

if (proc->time\_slice == 0 || proc->time\_slice > rq->max\_time\_slice) {

proc->time\_slice = rq->max\_time\_slice;

}

proc->rq = rq;

rq->proc\_num ++; //进程数加一

}

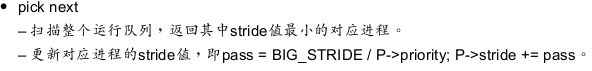


static void stride\_dequeue(struct run\_queue \*rq, struct proc\_struct \*proc) {

rq->lab6\_run\_pool=skew\_heap\_remove(rq->lab6\_run\_pool,&(proc->lab6\_run\_pool), proc\_stride\_comp\_f);

rq->proc\_num --;

}



static struct proc\_struct \*stride\_pick\_next(struct run\_queue \*rq) {

#if USE\_SKEW\_HEAP

if (rq->lab6\_run\_pool == NULL) return NULL;

struct proc\_struct \*p = le2proc(rq->lab6\_run\_pool, lab6\_run\_pool);

#else

list\_entry\_t \*le = list\_next(&(rq->run\_list));

if (le == &rq->run\_list)

return NULL;

struct proc\_struct \*p = le2proc(le, run\_link);

le = list\_next(le);

while (le != &rq->run\_list)

{

struct proc\_struct \*q = le2proc(le, run\_link);

if ((int32\_t)(p->lab6\_stride - q->lab6\_stride) > 0)

p = q;

le = list\_next(le);

}

#endif

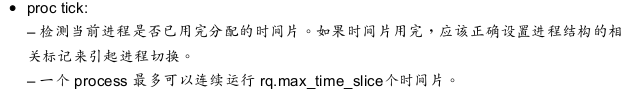
if (p->lab6\_priority == 0) //优先级为0

p->lab6\_stride += BIG\_STRIDE;

else p->lab6\_stride += BIG\_STRIDE / p->lab6\_priority;

return p;

}



static void stride\_proc\_tick(struct run\_queue \*rq, struct proc\_struct \*proc) {

if (proc->time\_slice > 0) { //到达时间片

proc->time\_slice --; //执行进程的时间片time\_slice减一

}

if (proc->time\_slice == 0) { //时间片为0

proc->need\_resched = 1; //设置此进程成员变量need\_resched标识为1,进程需要调度

}

}