spin 1000 &; spin 2000 &; spin 3000 &

sh 2 Put back into 2nd Queue

Program sh 2 has ran for 400 ms in 2nd Queue

sh 2 Enters Expired Queue

sh 2 Put back into 2nd Queue

Program sh 2 has ran for 300 ms in 2nd Queue

sh 3 Enters 1st Queue

Program sh 3 has ran for 100 ms in 1st Queue

Program sh 3 has ran for 300 ms in 2nd Queue

sh 4 Enters 1st Queue

Program sh 4 has ran for 100 ms in 1st Queue

Program sh 4 has ran for 400 ms in 2nd Queue

sh 4 Enters Expired Queue

sh 4 Put back into 2nd Queue

Program sh 4 has ran for 100 ms in 2nd Queue

sh 5 Enters 1st Queue

Program sh 5 has ran for 100 ms in 1st Queue

Program sh 5 has ran for 200 ms in 2nd Queue

Program sh 3 has ran for 100 ms in 2nd Queue

sh 3 Enters Expired Queue

sh 3 Put back into 2nd Queue

Program sh 3 has ran for 100 ms in 2nd Queue

sh 6 Enters 1st Queue

Program sh 6 has ran for 100 ms in 1st Queue

Program sh 5 has ran for 200 ms in 2nd Queue

sh 5 Enters Expired Queue

sh 5 Put back into 2nd Queue

Program sh 5 has ran for 100 ms in 2nd Queue

Program sh 5 has ran for 100 ms in 2nd Queue

Program sh 5 has ran for 100 ms in 2nd Queue

Program sh 6 has ran for 400 ms in 2nd Queue

sh 6 Enters Expired Queue

sh 7 Enters 1st Queue

Program sh 7 has ran for 100 ms in 1st Queue

Program sh 5 has ran for 100 ms in 2nd Queue

sh 5 Enters Expired Queue

sh 5 Put back into 2nd Queue

pid(5): x = 999

Program spin 5 has ran for 200 ms in 2nd Queue

pid(7): x = 1999

Program spin 7 has ran for 200 ms in 2nd Queue

zombie!

zombie!

Program init 1 has ran for 200 ms in 2nd Queue

Program sh 3 has ran for 300 ms in 2nd Queue

sh 3 Enters Expired Queue

sh 3 Put back into 2nd Queue

Program sh 3 has ran for 400 ms in 2nd Queue

sh 3 Enters Expired Queue

sh 3 Put back into 2nd Queue

Program sh 3 has ran for 100 ms in 2nd Queue

sh 8 Enters 1st Queue

$ Program sh 2 has ran for 100 ms in 2nd Queue

sh 2 Enters Expired Queue

Program sh 8 has ran for 100 ms in 1st Queue

Pid(8): x = 2999

Program spin 8 has ran for 400 ms in 2nd Queue

spin 8 Enters Expired Queue

spin 8 Put back into 2nd Queue

Program spin 8 has ran for 100 ms in 2nd Queue

Program init 1 has ran for 100 ms in 2nd Queue

init 1 Enters Expired Queue

init 1 Put back into 2nd Queue

zombie!

Program init 1 has ran for 100 ms in 2nd Queue

Side notes:

sh 2 was already started and was part of the start up process

Where the process enters the “Queue” that just means that its runtime number is either 0, 1, or 5 for 1st queue active queue and expired queue, respectively.

The code I included is only the part that I change other than allocproc. It sounds like you guys don’t need that one but let me know if you wanna see it. ( I literally only added 1 line of code to it)

Here is the Code:

scheduler(void)

{

struct proc \*p;

int unfnd = 0;

int sndfnd = 0;

int num\_of\_runs = 0;

for(;;){

// Enable interrupts on this processor.

sti();

// Loop over process table looking for process to run.

acquire(&ptable.lock);

unfnd = 0;

sndfnd = 0;

for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){

if(p->state != RUNNABLE)

continue;

if (p->rt == 0){

cprintf("%s ", p->name);

cprintf("%d ", p->pid);

cprintf("Enters 1st Queue \n");

proc = p;

switchuvm(p);

p->state = RUNNING;

swtch(&cpu->scheduler, proc->context);

switchkvm();

unfnd = 1;

p->rt++;

cprintf("Program %s %d has ran for 100 ms in 1st Queue \n", p->name, p->pid);

}

else if (p->rt < 5 && p->rt > 0 && unfnd==0)

{

/\* cprintf("This is second if ");

cprintf("%d ", unfnd);

cprintf("%d \n", sndfnd);\*/

while (p->rt < 5 && p->state == RUNNABLE)

{

/\*cprintf("%d ", p->rt);

cprintf("Secondary ");

cprintf("%s ", p->name);

cprintf("%d ", p->pid);

cprintf("%p\n", p);\*/

proc = p;

switchuvm(p);

p->state = RUNNING;

swtch(&cpu->scheduler, proc->context);

switchkvm();

p->rt++;

sndfnd = 1;

num\_of\_runs++;

}

cprintf("Program %s %d has ran for %d ms in 2nd Queue \n", p->name, p->pid, (num\_of\_runs)\* 100);

num\_of\_runs = 0;

if (p->rt >= 5 )

{

cprintf("%s ", p->name);

cprintf("%d ", p->pid);

cprintf("Enters Expired Queue \n");

}

}

else if (unfnd==0 && sndfnd==0)

{

if (p->state != RUNNABLE)

{

continue;

}

/\*cprintf("This is expired if ");

cprintf("%d ", unfnd);

cprintf("%d \n", sndfnd);

cprintf("%d ", p->rt);

cprintf("Expired ");

cprintf("%s ", p->name);

cprintf("%d ", p->pid);

cprintf("%p\n", p\*/

p->rt = p->rt - 4;

cprintf("%s ", p->name);

cprintf("%d ", p->pid);

cprintf("Put back into 2nd Queue \n");

}

proc = 0;

}

release(&ptable.lock);

}

}

Code from Header:

struct proc {

uint sz; // Size of process memory (bytes)

pde\_t\* pgdir; // Page table

char \*kstack; // Bottom of kernel stack for this process

enum procstate state; // Process state

int pid; // Process ID

struct proc \*parent; // Parent process

struct trapframe \*tf; // Trap frame for current syscall

struct context \*context; // swtch() here to run process

void \*chan; // If non-zero, sleeping on chan

int killed; // If non-zero, have been killed

struct file \*ofile[NOFILE]; // Open files

struct inode \*cwd; // Current directory

char name[16];

int rt; // Process name (debugging)

};