Lab2Answers

Haoran Lei

# 3.1 Monitoring CPU usage and waiting time of processes

```
#define IRQBASE          32        /* base ivec for IRQ0          */
```

This line was found in file “config/Configuration”. It defines the base interrupt vector for IRQ0 to be 32. Later in file clkint.c the following line was found:

```
set_evec(IRQBASE, (uint32)clkdisp);
```

The set_evec function is defined in evec.c, which sets exception vector to point to an exception handler, and in this case, sets exception 32 in IDT to be handled by clock interrupt dispatcher, which then calls clock interrupt handler.

# 3.2 and 3.3

In main(), I resume/created a process Lab2_test_fun_1 of higher priority which run arithmetic calculations and made systemcalls inside. There are several spots where I used kprintf systemcall (which is slow) to examine the relevant parameters of the two processes and validate the correctness of my code.

Before creating process” Lab2_test_fun_1”:

main() gains control of the cpu

miliseconds since boot: 4

currproctime is 5

gross ms of main() running on cpu is 7

start waiting time of main() is 1

pwaittime of main() is 0

pwaitcount of main() is 1

average waiting time of main() is 0

After creating process” Lab2_test_fun_1” but before it does calculations and make systemcalls

Lab2_test_fun_1() gains control of the cpu

miliseconds since boot: 26

currproctime is 7

gross ms of Lab2_test_fun_1() is 9

start waiting time of Lab2_test_fun_1() is 22

pwaittime of Lab2_test_fun_1() is 0

pwaitcount of Lab2_test_fun_1() is 1

average waiting time of Lab2_test_fun_1() is 0

gross ms of main() is 21

start waiting time of main() is 22

pwaitcount of main() is 2

(Doing calculations and making systemcalls)

this is a syscall

After it does calculations and make systemcalls:

after calculations and syscalls in Lab2_test_fun_1()

miliseconds since boot: 64

currproctime is 44

gross ms of Lab2_test_fun_1() is 46

start waiting time of Lab2_test_fun_1() is 22

pwaittime of Lab2_test_fun_1() is 0

pwaitcount of Lab2_test_fun_1() is 1

average waiting time of Lab2_test_fun_1() is 0

gross ms of main() is 21

start waiting time of main() is 22

pwaitcount of main() is 2

After the process terminates:

main() gains control of the cpu

miliseconds since boot: 97

currproctime is 5

gross ms of main() running on cpu is 28

start waiting time of main() is 22

pwaittime of main() is 72

pwaitcount of main() is 2

average waiting time of main() is 36

Lab2_test_fun_1()

Comments:

1) currproctime: this parameter is always positive and increasing as long as a process is not removed from cpu. Once it is, the parameter is reset to 0;
2) gross cpu usage: when a process is current, it keeps increasing, and once the process becomes ready, it freezes.
3) Start waiting time: this parameter for main stays the same both when Lab2_test_fun_1 is executing or main regains control of the cpu.
4) Waittime: the wait time of main is almost equal to the gross cpu usage of Lab2_test_fun_1 (…since Lab2_test_fun_1 makes too many kprintf calls which is a slow I/O call, the two could be different but always with the wait time greater than Lab2_test_fun_1's waiting time.)
5) Waitcount: main when created and resumed it is put to the ready list, so it starts to wait. When Lab2_test_fun_1 kicked main from cpu, the parameter incremented by one and stays the same after Lab2_test_fun_1 terminates.
6) Average waiting time: equals waittime/waitcount

Other findings: in Lab2_test_fun_1, some parameters change before and after Lab2_test_fun_1 makes systemcalls and do calculations. Some stays the same.

# 5.5

# Benchmark 1: eight CPU processes (LOOP1=10, LOOP2= 2000000)

The following are the result of running. The average waiting time and gross cpu usage of the 8 processes are almost equal to each other, with a little (a few ms) rising of average waiting time in the middle and followed by its decreasing due to the waiting order in the que (first process created run first and terminates first, and the processes created in the middle of the creating sequence suffers more waiting counts than those on the tails of the sequence), but overall they are the same.

pid 4

proctype 0

clktimemilli 3180

gross cpu usage 455 ms

average waiting time 143 ms

pid 5

proctype 0

clktimemilli 3390

gross cpu usage 455 ms

average waiting time 153 ms

pid 6

proctype 0

clktimemilli 3575

gross cpu usage 455 ms

average waiting time 160 ms

pid 7

proctype 0

clktimemilli 3735

gross cpu usage 455 ms

average waiting time 164 ms

pid 8

proctype 0

clktimemilli 3870

gross cpu usage 455 ms

average waiting time 166 ms

pid 9

proctype 0

clktimemilli 3979

gross cpu usage 455 ms

average waiting time 165 ms

pid 10

proctype 0

clktimemilli 4065

gross cpu usage 455 ms

average waiting time 162 ms

pid 11

proctype 0

clktimemilli 4125

gross cpu usage 455 ms

average waiting time 156 ms

# Benchmark 2: eight I/O processes (IOSLEEP=100, LOOP1=150, LOOP2= 50000)

The following are the result of running. The average waiting time and gross cpu usage of the 8 processes are almost equal. Compared with the CPU processes, they have much lower average waiting time and less gross cpu usage due to less LOOP2.

pid 4

proctype 1

clktimemilli 15187

gross cpu usage 154 ms

average waiting time 0 ms

pid 11

proctype 1

clktimemilli 15190

gross cpu usage 153 ms

average waiting time 0 ms

pid 5

proctype 1

clktimemilli 15244

gross cpu usage 153 ms

average waiting time 0 ms

pid 10

proctype 1

clktimemilli 15259

gross cpu usage 153 ms

average waiting time 0 ms

pid 9

proctype 1

clktimemilli 15268

gross cpu usage 153 ms

average waiting time 0 ms

pid 8

proctype 1

clktimemilli 15314

gross cpu usage 153 ms

average waiting time 1 ms

pid 7

proctype 1

clktimemilli 15317

gross cpu usage 154 ms

average waiting time 1 ms

pid 6

proctype 1

clktimemilli 15326

gross cpu usage 154 ms

average waiting time 1 ms

# Benchmark 3: four I/O processes and four CPU processes

The 4 CPU-bound processes are almost equal in their gross cpu time and average waiting time, and the same goes for the 4 I/O-bound processes. Between the two groups, CPU usage was higher for the CPU-bound processes but waiting time to be lower for I/O-bound processes. This means that our R3 scheduling is at work: although created at the same time (within the 25ms time slice in main(), all 8 processes will be created and resumed and until main() whose priority=20=INITPRIO gives up the cpu, all 8 processes now will start queuing), the 8 processes are given cpu resources in two  different ways. The cpu processes are able to consume its timeslices and finishes up its much larger computation tasks before having to wait for the I/O processes to terminate, while the I/O processes are fast in user response with 2ms of average waiting time.

pid 4

proctype 0

clktimemilli 2036

gross cpu usage 457 ms

average waiting time 75 ms

pid 5

proctype 0

clktimemilli 2054

gross cpu usage 457 ms

average waiting time 78 ms

pid 6

proctype 0

clktimemilli 2071

gross cpu usage 457 ms

average waiting time 79 ms

pid 7

proctype 0

clktimemilli 2267

gross cpu usage 457 ms

average waiting time 75 ms

pid 8

proctype 1

clktimemilli 15864

gross cpu usage 153 ms

average waiting time 2 ms

pid 9

proctype 1

clktimemilli 15963

gross cpu usage 153 ms

average waiting time 2 ms

pid 11

proctype 1

clktimemilli 16015

gross cpu usage 153 ms

average waiting time 2 ms

pid 10

proctype 1

clktimemilli 16040

gross cpu usage 153 ms

average waiting time 2 ms