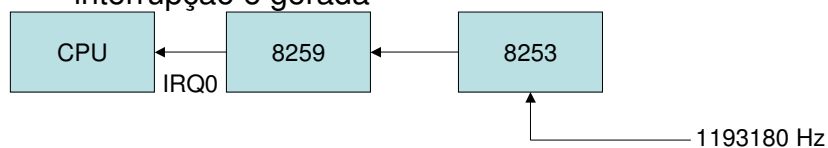


Escalonamento no LINUX

Configuração do TIMER

- Controla o intervalo entre interrupções do núcleo
- include/asm-i386/param.h
 - DEFAULT_HZ
 - HZ
- No Linux (depende do kernel HZ= 100), 10 ms
- Utilizado para configurar o 8253
 - $LATCH = (1193180 + HZ/2) / HZ$;
 - LATCH carrega o contador que é decrementado a cada 1.193180 Mhz, quando o contador chega a zero uma interrupção é gerada

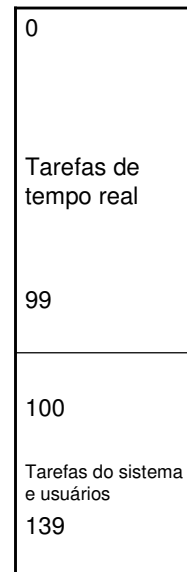


8253

- O LATCH é configurado no arquivo `/arch/i386/kernel/timers/timer_pit.c`
 - O valor do LATCH é armazenado em um registrador de 16 bits, portanto o valor máximo do contador é 65535, logo a menor frequência que poderá ser configurada é ~19 Hz
 - Portanto a maior frequência é 1,193180 Mhz
- ```
• outb_p(0x34,PIT_MODE); /* binary, mode 2, LSB/MSB, ch 0 */
• udelay(10);
• outb_p(LATCH & 0xff , PIT_CH0); /* LSB */
• udelay(10);
• outb(LATCH >> 8 , PIT_CH0); /* MSB */
```

## Escalonamento no Linux

- Múltiplas filas por prioridade
  - 140 níveis de prioridade
  - Menor valor -> maior prioridade
- Três políticas de escalonamento
  - SCHED\_OTHER= prioridades 100 a 139
  - SCHED\_FIFO = prioridades 0 a 99
  - SCHED\_RR= prioridades 0 a 99



## Escalonamento do Linux (2)

- Duas filas de processos prontos
  - Active
  - Expired
- O escalonador escolhe os processos de acordo com sua prioridade (maior prioridade primeiro), na fila de processos ativos (*active*)
- Quando o processo expira sua fatia de tempo, ele passa para fila *expired*, ou seja, o processo está pronto, mas não mais habilitado para executar
- Quando não existirem mais processos na fila *active*, todos os processos *expired*, são passados para a fila *active*

## TIMESLICE

- No escalonamento SCHED\_OTHER, cada processo ganha uma fatia de tempo para execução (*timeslice*), proporcional a sua prioridade
- No kernel do LINUX (kernel/sched.c)
  - #define MIN\_TIMESLICE max(5 \* HZ / 1000, 1)
  - #define DEF\_TIMESLICE (100 \* HZ / 1000)
  
  - DEF\_TIMESLICE = 10
    - Considerando o TIMER foi configurado para 100Hz (período de 10ms), o DEF\_TIMESLICE equivale a 100 ms

## TIMESLICE(2)

- Função que calcula o *timeslice*
  - `task_timeslice` (kernel/sched.c)
- Prioridade 100: 800 ms
- Prioridade 120: 100 ms( `DEF_TIMESLICE`)
- Prioridade 139: 5 ms (`MIN_TIMESLICE`)
- Além da prioridade estática, cada processo tem sua prioridade dinâmica. A política é baseada em bônus e penalidades
  - Bônus -5: processo altamente interativo
  - Penalidade +5: processo intensivo em uso da CPU
  - Calculado de acordo com o tempo em que o processo fica em estado de espera (campo `SLEEP_AVG` na estrutura de dados `task_struct`)

## Escalonamento

- Qual a influência da frequência utilizada no TIMER?
- Qual a influência da fatia de tempo (*timeslice*) ?
  - Aplicações em lote
  - Aplicações com interação com o usuário

## 1º. Experimento

- Redefina o TIMER para o menor número possível ou seja HZ e DEFAULT\_HZ = 19
- Recompile o kernel (não se esqueça de gerar o patch)
- Execute uma aplicação que crie 10 processos filhos e execute um laço relativamente grande
- Compare com o kernel original
  - número de trocas de contexto através do arquivo /proc/stat na variável ctxt
  - o tempo de execução
  - o tempo de resposta, através da utilização de aplicações e menus na interface gráfica

## 2º. Experimento

- Redefina o DEF\_TIMESLICE para 800
- Recompile o kernel
- Reinicialize a máquina
- Execute uma aplicação que crie 10 processos filhos e execute um laço relativamente grande
- Compare com o kernel original
  - número de trocas de contexto através do arquivo /proc/stat na variável ctxt
  - o tempo de execução
  - o tempo de resposta, através da utilização de aplicações e menus na interface gráfica