E/S direta no Linux

Interface Hardware-Software - 2005.1

Lauro Moura
lauromoura@gmail.com

Roteiro

- E/S direta em C
- Temporização
- Exemplo: Speaker

E/S direta em C - Introdução

- Funções de acesso (macros) definidas no arquivo **`<sys/io.h>`**
- Devido a limitações do gcc, é necessário o uso de otimização (**`-O1`**) durante a compilação.
- Exemplo:

```
$ gcc -O1 fonte.c
```

E/S direta em C - Permissões

- Antes de acessar as portas, é necessário dar permissão de acesso ao programa
- Funções **`ioperm()`** e **`iopl()`**
- Ambas necessitam de privilégio de super-usuário
- Razão: Usuários comuns são maus...

E/S direta em C - Permissões `ioperm()`

- Definida em **`<sys/io.h>`**
- Modifica a permissão de acesso de um processo para as portas especificadas
- Apenas as primeiras 0x3FF portas. Para outras portas é necessário **iopl**
- Processos filhos não herdam permissões.

E/S direta em C - Permissões `iopl()`

- Definida em **`<sys/io.h>`**
- Altera o nível de privilégio do processo (3 = privilégio total - root)
- Além do acesso a todas a portas, pode desabilitar interrupções
- Processos filhos herdam permissões.
- Use apenas se necessário

E/S direta em C - Acesso `in*()` e `out*()`

- Macros definidas em **`<sys/io.h>`**
- Acessam diretamente as portas
- Sufixos:
  - **`inb()`, `outb()`** – byte (8 bits) – Mais usadas
  - **`inw()`, `outw()`** – word (16 bits)
- As macros que usam word acessam duas portas por vez: Porta x e x+1

E/S direta em C - Acesso `in*()` e `out*()`

- in* recebe como argumento o número da porta e retorna o resultado (byte ou word)
- out* recebe como argumento o valor a ser enviado e a porta (nesta ordem).

E/S direta em C - Acesso `in*_p()` e `out*_p()`

- Também definidas em **`<sys/io.h>`**
- Diferem por esperar a E/S completar.
- Mesmos sufixos das versões normais.

Temporização

- Devido à natureza multi-tarefa, não há controle **exato** de temporização.
- Um processo pode ficar parado de 10 milissegundos a alguns segundos.
- Apenas tarefas especiais precisam de um controle maior do tempo.

Temporização
Sleeping / Delays

- Funções **`sleep()`** e **`usleep()`**
- Pausam a execução por alguns segundos (**`sleep`**) ou alguns milissegundos (**`usleep`**, aparentemente 10 no mínimo), liberando a CPU
- Para delays menores que 50 milissegundos pode não ser vantajoso liberar a CPU, devido à troca de contexto.

Temporização – Medição de Tempo

- **`<time.h> - time()`** - Tempo em segundos desde 1º de Janeiro de 1970.
- **`<sys/time.h> - gettimeofday()`** - Igual a time() mas com precisão de microssegundos ou segundos.

Exemplo: PIT 8253/8254 (Speaker)

- *Programmable Interval Timer*
- Composto por 3 contadores, também chamados de canais, mais um registrador de controle.
- Cada contador é independente dos outros.
- Código: speaker.c

PIT 8253
Funcionamento Básico

- Controla-se 1 canal por vez.
- É enviada uma palavra de controle com o modo de execução para o registrador de controle (porta 0x43).
- Logo em seguida os dados são enviados para o contador correspondente (0x40 até 0x42).
- Speaker é controlado pela porta 0x42

PIT 8253
Funcionamento Básico (cont.)

- Frequência básica de 1.19318MHz
- Para obter a frequência efetiva, o contador utiliza um valor de 2 bytes (que você deve definir) para dividir essa frequência real.
- Frequências
  - Máxima (div = 1) 1.19318MHz
  - Mínima (div = 65535) 18.2 Hz
- Origem nas frequências NTSC

PIT 8253
Palavra de Controle - 0x43

- Bits 7,6 – Seleção do contador
- Bits 5,4 – Ordem de carregamento
- Bits 3-1 – Modo de operação
- Bit 0 – BCD/Binário
- Mais detalhes -> Referências

PIT 8253
Exemplo: Speaker

- Palavra de controle – 1011x110
- 10 – Contador 0x42
- 11 – Carregar primeiro LSB, depois o MSB
- x11 – Gerar onda quadrada
- 0 – Contagem binária

PIT 8253
Speaker: Ativação

- Bits 1 e 0 lidos da porta 0x61 devem estar ligados.
- Para desativar basta desligar ambos.

Referências

- Man pages
- Linux I/O port programming mini-HOWTO
- Pdf's em www.cin.ufpe.br/~lmmn/ihs/