Multiplexagem Síncrona de E/S

ProgA

Patricio Domingues

Multiplexagem de E/S

- Conceito de *multiplexagem*
- Como estar atento a vários eventos? http://bit.ly/2j7Hvd1
  - Exemplo
    - Aguardar SIMULTANEAMENTE por conteúdo
      - teclado (stdin)
      - vários sockets (um por ligação)
  - Exemplo – serviço distribuído de *chat*
    - Servidor
      - Sempre que um cliente escreve, o conteúdo deve ser encaminhado para a outra entidade comunicante (ou as outras, se for broadcast)
    - Cliente
      - Deve estar atento a conteúdo do teclado (utilizador escreve) e a conteúdo dos sockets (conversas em que está envolvido)

Função *select (#1)*

- `int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`
  - Função que permite determinar o estado de vários descritores

- Pode ser configurada para aguardar por eventos em vários descritores (ficheiros, sockets)
  - Retorna quando ocorre um evento ou quando expirou o temporizador (parâmetro `timeout`)
- Configurada com vetores de descritores para verificar…
  - Leitura: parâmetro `readfds`
  - Escrita: parâmetro `writefds`
  - Exceção: parâmetro `exceptfds`
- nfds
  - MAX(nº de elementos dos vetores readfs, writefds e exceptfds)

Função select (#2)

- Conjunto de descritores
  - Tipo de dado `fd_set`
  - Vetor de inteiros
    - Cada bit em cada inteiro correponde a um descriptor
  - 4 macros para uso com conjunto de descritores

```
void FD_CLR(int d, fd_set *set); // inibe bit para descritor d
void FD_SET(int d, fd_set *set); // ativa bit para descritor d
int  FD_ISSET(int d, fd_set *set);// bit d está ativo?
void FD_ZERO(fd_set *set);    // limpa bits do conjunto
```

Exemplo de uso de fd_set

```
int sockUDP1 = socket(…);
int sockUDP2 = socket(…);

fd_set select_set;
FD_ZERO(&select_set);   // inicializa conjunto
FD_SET(0,&select_set);  // ativa para STDIN
FD_SET(sockUDP1,&select_set); // ativa sockUDP1
FD_SET(sockUDP2,&select_set); // ativa sockUDP2
```

Função *select (#3)*

- `int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`
  - Parâmetro nfds
    - Especifica o número de descritores a serem testados
    - Deve ser igual ao valor do maior descritor mais um
    - Exemplo: descritores 3,5 e 9
      - nfds deve ser 10
      - Código deve calcular nfds

- A constant FD_SETSIZE indica qual é o valor máximo suportado pelo Sistema
- Exemplo

```
#include <stdio.h>
#include <sys/select.h>
int main(void){
        printf("FD_SETSIZE=%d\n", FD_SETSIZE);
        return 0;
}
```

```
FD_SETSIZE=1024
```

Função *select (#4)*

- `int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`
  - Os parâmetros readfds, writefds e exceptfds são parâmetros **valores/resultados**
    - São inicializados com os descritores que se pretendem monitorizar
    - Quando a função retorna, os vetores indicam quais os descritores que registaram atividade
- Modus operandi (ciclo)
  - Definir os descritores que se pretendem monitorizar
  - Carregar os vetores readfds, writefds e exceptfds
    - Ativar os descritores pretendidos
  - Definir o temporizador
  - Chamar a função
  - Quando a função retorna, determinar o que se passou
    - Quantos descritores estão ativos
      - Zero descritores = timeout
    - Que descritores estão *ativos*?
    - Deteção é feita com a macro FD_ISSET(descritor, set)

Temporizador

- `int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`
- Estrutura struct timeval

```
struct timeval {
    long    tv_sec;   // seconds
    long    tv_usec; // microseconds
};
```

- A estrutura `struct timeval` tem que ser reiniciada antes de cada chamada
- Caso não se pretenda temporizador (*espera bloqueante*), deve ser passado NULL no parâmetro de temporização
- A função select devolve zero quando o temporizador expira…

Valor devolvido por select

- `int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`

- A função select devolve
  - -1 quando ocorre erro do sistema ou é recebido signal
    - É atribuído um código de erro à variável errno
      - EINTR no caso de signal a interromper o select
  - Nº de descritores ativos
  - Zero quando o temporizador expira

Descritor ativo

- Condições que desencadeiam o estado ativo de um descritor

Condição	Leitura?	Escrita?	Exceção?
Dados para ler	Sim		
Parte de leitura da ligação foi fechada	Sim		
Novo ligação (socket de escuta)	Sim		
Espaço para escrita		Sim	
Parte escrita da ligação foi fechada		Sim	
Erros pendentes	Sim	Sim	
Dados *out of band*			Sim

Função select – casos de uso

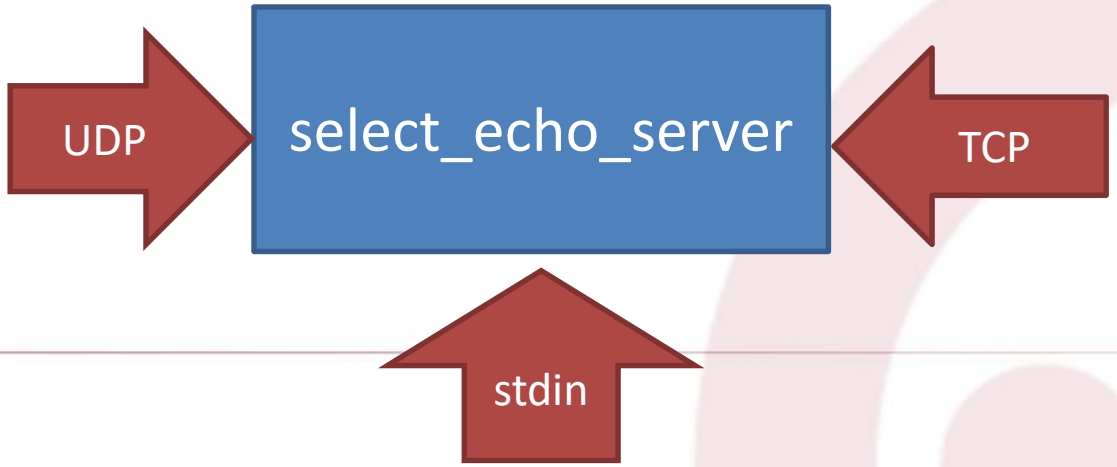
- Espera indefinida
  - Função apenas desbloqueia quando ocorrer evento
- Espera finita
  - Função desbloqueia quando:
    - ocorre evento

    OU
    - temporizador expira
- Sem espera
  - Temporizador a zero
  - Função verifica descritores e retorna imediatamente

- `select`
  - Chamada bloqueante
  - Pode ser interrompida por um signal
    - Select devolve -1 e errno fica com o valor EINTR

select: exemplo

- select_echo_server
  - Servidor de echo que processa eventos de três entradas distintas
    - stdin (teclado)
    - socket TCP no porto *N*
    - socket UDP no porto *N*
  - Sempre que deteta um evento, processa o evento, efetuando o echo do conteúdo



select_echo_server

- Ficheiros
  - server.c (server.c.html)
  - common.h (common.h.html)
  - common.c (common.c.html)
- Utilitário network cat (nc) é empregue como cliente TCP e UDP
  - Cliente TCP

    nc -t 127.0.0.1 porto
  - Cliente UDP

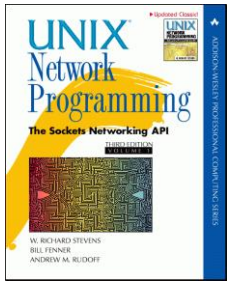
    nc -u 127.0.0.1 porto

pselect

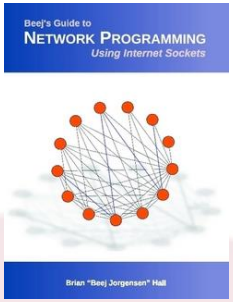
- `int pselect(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, const struct timespec *timeout, const sigset_t *sigmask);`
- Versão POSIX de select
  - Comportamento similar
- Principais diferenças `pselect` vs. `select`
  - Temporizador: `struct timespec` em lugar de `struct timeval`
    - Precisão da ordem nanosegundo
  - Temporizador: `pselect` nunca altera o valor da estrutura struct timespec (const)
    - `select` pode alterar esse valor
  - `pselect` acrescenta parâmetro sigmask que possibilita bloquear *signals*

Bibliografia

- Leitura recomendada
  - *UNIX Network Programming, Volume 1, 3rd edition: Networking APIs: Sockets and XTI*, Prentice Hall, 2003, 978-0131411555.
    - Section 6 / . I/O Multiplexing: The select and poll Functions



  - *Beej's Guide to Network Programming - Using Internet Sockets*, Brian "Beej Jorgensen" Hall, 2016 (http://beej.us/guide/bgnet/)



  - `man select`
  - `man 7 time`