Guia de utilização do `select()`

Este guia segue de perto a informação disponibilizada em Linux na página do manual relativa à chamada de sistema `select()`

J.Sanguino
Fevereiro de 2024

```
SYNOPSIS
#include <sys/select.h>

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

Breve Descrição

A chamada de sistema `select()` permite que um programa monitorize um conjunto de descritores até que pelo menos um esteja pronto para leitura, ou pronto para escrita, ou tenha uma condição de exceção assinalada.

Os descritores a monitorizar são especificados através dos argumentos da chamada de sistema. No retorno do `select()` os mesmos argumentos são utilizados para identificar os descritores que causaram a terminação da chamada de sistema.

Se nenhum dos descritores especificados estiver pronto para a classe de eventos especificada (pronto para leitura, ou pronto para escrita, ou tenha uma condição de exceção assinalada) o `select()` irá bloquear, a menos que um *timeout* tenha sido especificado e este expire.

Guia de utilização do `select()`

Motivação para a utilização do `select()`

A utilização desta chamada de sistema é particularmente interessante quando um programa aguarda *input* simultaneamente por mais do que um descritor e não pode correr o risco de bloquear, aguardando que um descritor fique pronto para leitura, deixando de ler os restantes descritores.

Por exemplo, um programa que aguarde *input* do `stdin` e simultaneamente aguarde *input* de um *socket* de UDP,

- não pode usar a função `fgets()`, para ler do `stdin`, sem estar seguro de que o `stdin` está pronto para leitura, ou corre o risco de bloquear até que o `stdin` esteja pronto para leitura, não tendo, enquanto estiver bloqueado no `fgets()`, hipótese de ler o *socket* UDP (se o `stdin` não ficar pronto para leitura, o programa permanecerá bloqueado no `fgets()` e não irá ler o *socket* UDP);
- não pode usar a chamada de sistema `recvfrom()`, para ler do *socket* UDP, sem estar seguro de que o *socket* está pronto para leitura, ou corre o risco de bloquear até que o *socket* esteja pronto para leitura, não tendo, enquanto estiver bloqueado no `recvfrom()`, hipótese de ler o `stdin` (se o *socket* não ficar pronto para leitura, o programa permanecerá bloqueado no `recvfrom()` e não irá ler o `stdin`);
- pode usar a chamada de sistema `select()`, especificando que os descritores do `stdin` (descritor `0`) e do *socket* UDP devem ser monitorizados até estarem prontos para leitura; assim que um dos descritores esteja pronto para leitura o `select()` terminará indicando quais os descritores que estão prontos para leitura; a leitura desses descritores poderá agora ser feita com a garantia que o programa não irá bloquear na leitura.

Existem alternativas à utilização da chamada de sistema `select()` que não são abordadas neste guia.

Guia de utilização do `select()`

Especificação da interface da chamada de sistema `select()`

O protótipo da chamada de sistema `select()` é o seguinte:

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

em que,

`nfds` – é o número do maior descritor a monitorizar, mais um;

`readfds`, `writefds`, `exceptfds`
- endereço de variáveis do tipo `fd_set`, previamente preenchidas com a especificação dos descritores a monitorizar para leitura, para escrita e por condição de exceção, respetivamente, e que na terminação da chamada de sistema indicarão os descritores prontos para leitura, prontos para escrita, e que tenham uma condição de exceção assinalada;
- estes três argumentos são usados como argumentos de entrada e como argumentos de saída;
- qualquer um destes três argumentos pode ser especificado como `NULL` se não houver descritores para monitorizar a classe de eventos correspondente (pronto para leitura, ou pronto para escrita, ou tenha uma condição de exceção assinalada);

`timeout`
- endereço de estrutura do tipo `timeval` com a especificação da duração máxima do intervalo de tempo que o `select()` deve bloquear enquanto monitoriza os descritores especificados;
- este argumento pode ser especificado como `NULL`, situação em que o `select()` irá bloquear até que um dos descritores a monitorizar fique pronto, de acordo com a classe de eventos especificada (pronto para leitura, ou pronto para escrita, ou tenha uma condição de exceção assinalada).

Guia de utilização do `select()`

Especificação dos descritores a monitorizar

A especificação dos descritores a monitorizar pelo `select()` é feita através do preenchimento de variáveis do tipo `fd_set`. Cada variável `fd_set` representa um conjunto de descritores. Esse conjunto pode ser manipulado através das seguintes macros:

`void FD_ZERO(fd_set *set);`	A macro `FD_ZERO()` remove todos os descritores do conjunto cujo endereço é `set`. Deve ser utilizada para inicializar um conjunto de descritores.
`void FD_SET(int fd, fd_set *set);`	A macro `FD_SET()` adiciona o descritor `fd` ao conjunto de descritores cujo endereço é `set`.
`void FD_CLR(int fd, fd_set *set);`	A macro `FD_CLR()` remove o descritor `fd` do conjunto de descritores cujo endereço é `set`.

Na chamada do `select()` os descritores incluídos em cada conjunto serão monitorizados de acordo com a classe de eventos correspondente (pronto para leitura, ou pronto para escrita, ou tenha uma condição de exceção assinalada).

Na terminação do `select()` os descritores incluídos em cada conjunto identificam os descritores responsáveis pela terminação do `select()`.

As mesmas variáveis que são utilizadas como argumento de entrada para especificar os descritores a monitorizar pelo `select()` são utilizadas (alteradas) como argumento de saída para identificar os descritores responsáveis pela terminação do `select()`.

Após a execução do `select()`, a macro

```
int  FD_ISSET(int fd, fd_set *set);
```

pode ser utilizada para verificar se um determinado descritor `fd` está incluído no conjunto cujo endereço é `set`. A macro retorna zero se o descritor não estiver incluído no conjunto e retorna diferente de zero no caso contrário.

Guia de utilização do `select()`

Especificação do `timeout`

A especificação de um timeout no `select()` é feita através da seguinte estrutura `timeval`:

```
struct timeval {
               time_t      tv_sec;         /* seconds */
               suseconds_t tv_usec;        /* microseconds */
               };
```

Retorno do `select()`

Após a correta execução o `select()` retorna o número de descritores responsáveis pela sua terminação e identificados nos argumentos relativos aos conjuntos de descritores monitorizados. O retorno pode ser zero se a terminação tiver sido causada pelo temporizador.

Em caso de erro o `select()` retorna -1 e a variável global `errno` é preenchida com informação mais detalhada acerca do erro ocorrido.

Mais informação…

… acerca do `select()` pode ser encontrada na página do manual relativa a esta chamada de sistema. Para aceder a essa informação num sistema Linux, entrar na linha de comandos de um terminal o comando: `man select`.

Guia de utilização do `select()`

Exemplo 1:

Considere-se o caso de uma aplicação em que se criou um *socket* UDP, que tem associado o descritor `fd`, e de onde se aguarda *input*, mas também se aguarda, em simultâneo, *input* do `stdin` (descritor `0`).
A utilização da chamada de sistema `recvfrom()` no *socket* `fd` tem o risco de o programa ficar bloqueado no `recvfrom()`, enquanto não chegar nenhuma mensagem.
Da mesma forma, a utilização da função `fgets()`, para ler do `stdin` (descritor `0`), tem o risco de o programa ficar bloqueado no `fgets()`, enquanto o `stdin` não tiver *input*.
Uma possível solução é a utilização de um `select()` para monitorizar simultaneamente o descritor `0` (`stdin`) e o descritor `fd` (*socket* UDP) e aguardar (bloquear) até que um dos descritores esteja pronto para leitura. Neste contexto, o bloqueio é perfeitamente aceitável, assumindo que a aplicação não tem tarefas a executar enquanto não chegar *input* do `stdin`, ou do *socket*.

Na chamada do `select()` as variáveis `maxfd`, do tipo `int`, e `rfds`, do tipo `fd_set`, serão utilizadas como argumentos e previamente inicializadas da seguinte forma:

```
maxfd=fd; /* fd é o maior dos descritores a monitorizar, dado que o descritor do stdin é 0 (menor que fd) */

FD_ZERO(&rfds);   /* remover todos os descritores do conjunto */
FD_SET(0,&rfds);  /* adicionar o descritor 0 (stdin) ao conjunto */
FD_SET(fd,&rfds); /* adicionar o descritor fd (socket UDP) ao conjunto */
```

A chamada do `select()` assumirá assim a seguinte forma:

```
counter=select(maxfd+1,&rfds,(fd_set*)NULL,(fd_set*)NULL,(struct timeval*)NULL);
```

Ao executar o `select()` o programa irá bloquear até que um dos descritores esteja pronto para leitura.

Quando um dos descritores ficar pronto para leitura (e podem ficar simultaneamente os dois),
o `select()` irá retornar, para a variável `counter`, do tipo `int`, o número de descritores prontos para leitura;
`FD_ISSET(0,&rfds)`, se posteriormente executada, retornará diferente de zero se o descritor `0` (`stdin`) estiver pronto para leitura e zero, no caso contrário;
`FD_ISSET(fd,&rfds)`, se posteriormente executada, retornará diferente de zero se o descritor `fd` (*socket* UDP) estiver pronto para leitura e zero, no caso contrário.

No caso de se querer implementar um ciclo, por forma a aguardar por mais *input*, há que ter em conta que o argumento `rfds` tem de ser reinicializado antes de cada chamada do `select()`, uma vez que este pode ter sido alterado na anterior execução do `select()`. O argumento `rfds` é argumento de entrada e de saída.

Note-se que, na sequência dos passos anteriores, o descritor que for detetado como pronto para leitura deve ser lido antes que um novo `select()` seja executado para o monitorizar, nos mesmos moldes. Caso contrário, o novo `select()` irá retornar imediatamente por o descritor continuar pronto para leitura.

Guia de utilização do `select()`

Exemplo 2:

Suponha-se o caso de uma aplicação em que se criou um *socket* TCP, que tem associado o descritor `fd`, colocado em modo de escuta (*listen*), e onde se aguarda o estabelecimento de sessões por iniciativa de clientes TCP remotos, mas a mesma aplicação também aguarda, em simultâneo, *input* do `stdin` (descritor `0`).
A utilização da chamada de sistema `accept()` no *socket* `fd` tem o risco de o programa ficar bloqueado no `accept()`, enquanto nenhum cliente TCP fizer `connect()`.
Da mesma forma, a utilização da função `fgets()`, para ler do `stdin` (descritor `0`), tem o risco de o programa ficar bloqueado no `fgets()`, enquanto o `stdin` não tiver *input*.
Uma possível solução é a utilização de um `select()` para monitorizar simultaneamente o descritor `0` (`stdin`) e o descritor `fd` (*socket* TCP) e aguardar (bloquear) até que um dos descritores esteja pronto para leitura. Neste contexto, o bloqueio é perfeitamente aceitável, assumindo que a aplicação não tem tarefas a executar enquanto não chegar *input* do `stdin`, ou do *socket*.

Na chamada do `select()` as variáveis `maxfd`, do tipo `int`, e `rfds`, do tipo `fd_set`, serão utilizadas como argumentos e previamente inicializadas da seguinte forma:

```
maxfd=fd; /* fd é o maior dos descritores a monitorizar, dado que o descritor do stdin é 0 (menor que fd) */

FD_ZERO(&rfds);   /* remover todos os descritores do conjunto */
FD_SET(0,&rfds);  /* adicionar o descritor 0 (stdin) ao conjunto */
FD_SET(fd,&rfds); /* adicionar o descritor fd (socket TCP) ao conjunto */
```

A chamada do `select()` assumirá assim a seguinte forma:

```
counter=select(maxfd+1,&rfds,(fd_set*)NULL,(fd_set*)NULL,(struct timeval*)NULL);
```

Ao executar o `select()` o programa irá bloquear até que um dos descritores esteja pronto para leitura.

Quando um dos descritores ficar pronto para leitura (e podem ficar simultaneamente os dois),
o `select()` irá retornar, para a variável `counter`, do tipo `int`, o número de descritores prontos para leitura;
`FD_ISSET(0,&rfds)`, se posteriormente executada, retornará diferente de zero se o descritor `0` (`stdin`) estiver pronto para leitura e zero, no caso contrário;
`FD_ISSET(fd,&rfds)`, se posteriormente executada, retornará diferente de zero se o descritor `fd` (*socket* TCP) estiver pronto para leitura e zero, no caso contrário.

Guia de utilização do `select()`

Exemplo 2 (continuação):

Estando o descritor `fd` associado a um servidor TCP, ao ser assinalado como estando pronto para leitura, a aplicação deverá executar a chamada de sistema `accept()`, a qual irá retornar um novo descritor, `newfd`, do tipo `int`, que irá estar associado à nova sessão TCP agora estabelecida.

Na próxima chamada ao `select()`, para aguardar por mais *input*, há que ter em atenção que agora existirá mais um descritor a monitorizar (`newfd`), o que deve ser tido em conta na inicialização da variável `maxfd` (maior dos descritores `0`, `fd` e `newfd`) e na inicialização da variável `rfds` (necessário adicionar `newfd` ao conjunto dos descritores a monitorizar; `FD_SET(newfd,&rfds);`).

Assim, numa nova chamada ao `select()` três descritores serão agora monitorizados (`0`, `fd` e `newfd`).

No próximo retorno do `select()` a macro `FD_ISSET()` será agora usada para identificar quais dos três descritores (`0`, `fd` e `newfd`) estão prontos para leitura.

Realçar que a aplicação descrita foi iniciada a monitorizar dois descritores (`0` e `fd`) e depois de receber um pedido de estabelecimento de sessão passou a monitorizar três descritores (`0`, `fd` e `newfd`). No caso desta aplicação, será natural que o número de descritores a monitorizar varie ao longo tempo dependendo dos pedidos de estabelecimento de sessão que forem recebidos, o que trás um acréscimo de um descritor por cada sessão adicional, e depende também das sessões que entretanto forem terminadas, o que trás um decréscimo de um descritor por cada sessão terminada.

Para uma aplicação genérica, os descritores a monitorizar pelo `select()` num determinado instante dependerão do estado da aplicação nesse instante.

Guia de utilização do `select()`

Exemplo 3:

Suponha-se o caso em que se criou um *socket* UDP, que tem associado o descritor `fd`, e através desse *socket* foi enviada uma mensagem para um servidor UDP e se aguarda resposta.
A utilização da chamada de sistema `recvfrom()` neste *socket* tem o risco de, no caso da resposta do servidor não chegar, o programa ficar bloqueado no `recvfrom()`.
Uma possível solução é a utilização de um `select()` para aguardar que o *socket* fique pronto para leitura, mas com um temporizador, e só executar o `recvfrom()` se o *socket* ficar pronto para leitura.

Na chamada do `select()` as variáveis `maxfd`, do tipo `int`, e `rfds`, do tipo `fd_set`, e `tout`, do tipo `struct timeval`, serão utilizadas como argumentos e previamente inicializadas da seguinte forma:

```
maxfd=fd; /* fd é o único descritor a monitorizar */

FD_ZERO(&rfds);   /* remover todos os descritores do conjunto */
FD_SET(fd,&rfds); /* adicionar o descritor fd ao conjunto */

tout.tv_sec=2; /* 2 segundos, como exemplo */
tout.tv_usec=0;
```

A chamada do `select()` assumirá assim a seguinte forma:

```
counter=select(maxfd+1,&rfds,(fd_set*)NULL,(fd_set*)NULL,&tout);
```

Ao executar o `select()` o programa irá bloquear até que um de dois eventos ocorra:

i) o descritor `fd` fica pronto para leitura (receção de uma mensagem); o `select()` retornará 1 (há um descritor pronto para leitura); `FD_ISSET(fd,&rfds)`, se posteriormente executada, retornará diferente de zero (o descritor `fd` está incluído no conjunto `rfds`);

ii) o temporizador expira; o `select()` retornará zero (terminação causada pelo temporizador); `FD_ISSET(fd,&rfds)`, se posteriormente executada, retornará zero (o descritor `fd` não está incluído no conjunto `rfds`).

No caso de se querer implementar um ciclo, por forma a retransmitir a anterior mensagem um determinado número de vezes, há que ter em conta que os argumentos `rfds` e `tout` têm de ser reinicializados antes de cada chamada do `select()`, uma vez que estes podem ter sido alterados na anterior execução do `select()`.