EA872 Laboratório de Programação de Software Básico
Atividade 9

Vinícius Esperança Mantovani **RA 247395**

Entrega (limite): 04/10/2023, em aula.

Atividade 3)

1)

A seguir, apresenta-se o código comentado, seguido imediatamente pela explicação a respeito dele:

```
/* Programa teste_select.c */
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int main(void) {
 int n, c; /* n eh usado para coletar o retorno de select e c para
printar o primeiro caracter recebido do input padrao*/
 long int tolerancia = 5; /* define o tempo maximo que select pode
levar durante a verificacao*/
 fd_set fds; /* define um conjunto de file descriptors, chamado de
fds*/
 struct timeval timeout; /* define um objeto da estrutura timeval, para
representacao temporal*/
 int fd; /* define um inteiro que sera usado como file descriptor*/
  fd = 0;              /* Esta linha seta o valor de fd como 0,
associado ao stdin*/
 FD_ZERO(&fds);       /* Remove todos os file descriptors do conjunto
fds */
```

```
 FD_SET(fd, &fds);    /* Adiciona o file descriptor fd ao conjunto fds
*/
 timeout.tv_sec = tolerancia;  /* Seta o valor de segundos de timeout
*/
 timeout.tv_usec = 0; /* seta o valor de microssegundos de timeout */

 n = select(1, &fds, (fd_set *)0, (fd_set *)0, &timeout);
 /* O que faz a linha acima?
 * A linha acima verifica se o primeiro file descriptor de fds esta
pronto para leitura, caso nao esteja, o file descriptor sera tirado de
fds;
 */

 if(n > 0 && FD_ISSET(fd, &fds))   /* esta linha verifica se existe
algum file descriptor de fds (o primeiro neste caso) pronto para
leitura e se o file descriptor fd ainda está presente no conjunto fds
de file descriptors apos a chamada select */
   {
     c = getchar();  /* O que acontecerá de diferente aqui? - O
primeiro caracter escrito em stdin sera lido pelo programa e salvo em
c*/
     printf("Caractere %c teclado.\n", c); /* printa o primeiro
caracter lido de stdin*/
   }
 else if(n == 0)    /* Aqui, caso o tempo de tolerancia especificado em
select seja terminado sem stdin passar conteudo para leitura, executa o
bloco a seguir */
       {
       printf("Nada foi teclado em %ld s.\n", tolerancia); /* printa um
aviso de que stdin nao tem nada a ser lido*/
       }
      else perror("Erro em select():");   /* Caso o retorno de select
seja -1, temos um erro, logo, printa que houve um erro neste caso */
 exit(1);   /* encerra o programa */
}
```

Para compreender satisfatoriamente o que o programa acima faz, vale analisá-lo linha por linha, conforme se faz a seguir.

De início, são declaradas algumas variáveis que serão definidas, algumas, mais adiante no código e, tolerância, que já é definida logo na declaração. As variáveis “n” e “c” são usadas, respectivamente, para conterem o retorno da função select e o primeiro caractere escrito em stdin. A variável “tolerancia” contém o valor em segundos que será usados como tempo máximo para verificação pela chamada “select”. Já “fd” e “timeout”, contêm, respectivamente, o valor do file descriptor que será colocado no set de file descriptors “fds” e, o objeto da estrutura “timeval” que é

passado como parâmetro em “select” para definir o tempo máximo para a verificação como sendo igual ao valor de “tolerancia”, cujo valor é salvo no campo “tv_sec” de “timeout”.

Seguindo pelo código, temos uma chamada de “select(1, &fds, (fd_set *)0, (fd_set *)0, &timeout)”, em que o primeiro parâmetro (1) define o número de file descriptors de “fds” a serem verificados por “select”, o segundo parâmetro (&fds) determina o conjunto de files descriptors que terá alguns ou todos de seus file descriptors verificados a respeito da prontidão para leitura, ou seja, a chamada “select” verificará se tais file descriptor estão prontos para leitura (contêm algo a ser lido). Em seguida, temos dois argumentos iguais “(fd_set *)0”, que determinam os conjuntos de file descriptors que serão analisados a respeito da prontidão para escrita e outras exceções, neste caso, sendo estados como o conjunto “0”, mostrando que tal verificação não é relevante no código. Por fim, temos como argumento “&timeout”, que define o tempo máximo durante o qual a chamada “select” espera que algum caracter esteja disponível para leitura no file descriptor “fd”, ou seja, no standard input (stdin).

Por fim, temos um bloco de condicionais que verifica o valor de retorno de “select” e se o file descriptor “fd” ainda está em “fds” depois da chamada de “select”. Para tanto, existe um “if” para o caso em que o retorno é maior que 0, o que representa que o “fd” está pronto para leitura, neste caso, representa que stdin tem informações a serem lidas e, além disso, verifica se “fd” ainda faz parte de “fds”, pois a chamada “select”, do modo como foi chamada, tira do conjunto os file descriptors que não estão prontos para leitura. Isso porque, assim se garante que “stdin” tem conteúdo a ser lido e, entra-se no bloco de tal condicional, no qual é escrito em “stdout” o primeiro caractere lido de “stdin”. A seguir, tem-se um “else if” que verifica se o retorno de select foi igual a 0, o que indicaria que não há nada a ser lido de “fd” (stdin), ou seja, ele não está pronto para leitura. Entrando nesse bloco, tem-se a impressão em “stdout” de uma mensagem avisando que não foi teclado nada (logo não há o que ser lido de “stdin”). Por fim, tem-se um “else” para o caso em que tivemos um erro em “select” (n < 0), neste caso, imprime-se em “stdout” uma mensagem avisando que houve erro em “select”.

De maneira sintetizada, percebe-se que o código funciona como um leitor do input padrão do sistema, escrevendo como retorno o primeiro caractere escrito nesse input, uma mensagem de erro se houve erro na leitura ou uma mensagem de timeout para o caso em que a verificação durou 5 segundos e o input ainda não tem nada para ser lido, ou seja, nada foi teclado.

2)

Isso não seria possível, pois embora seja possível definir diretamente o valor de “timeout.tv_sec” com aquilo que desejamos sem necessidade de usar a variável “tolerância”, ocorre que, ao tentarmos usar o valor desse campo diretamente no “printf”, temos o problema de o valor imprimido ser sempre 0. Isso ocorre porque o campo em questão é alterado durante a execução da chamada “select” e, seu valor é atualizado nesta chamada para conter o tempo restante no caso de ser teclado algo ou seu valor é definido como zero se a verificação terminar por conta de o tempo decorrido ser igual ao tempo definido inicialmente por “timeout”. Desse modo, acontece que, ao final da execução de “select”, o valor do campo “tv_sec” de “timeout” será diferente daquele definido inicialmente.

Teste executado:

```
Unset
pininsu@debian:~/Documents/EA872/ativ9$ ./teste_select
Nada foi teclado em 0 s.
```