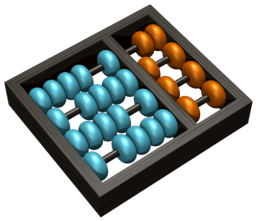


Exercício 6

Aluno: Felipe S. P. Carvalho e Arthur Maia Mendes

RA: 146040 e 135013

Instituto de Computação

Universidade Estadual de Campinas

Campinas, 22 de Outubro de 2018.

Sumário

1 Questão 1

Modificamos o programa cliente de echo do exercício 5 seguindo as especificações do enunciado e utilizamos a função `select` [1] para o envio de dados ao servidor. O seguinte trecho de código ilustra as alterações realizadas no programa cliente:

```
...
void doit(FILE *fp_in, int sockfd) {
    ...
    stdineof = 0;
    FD_ZERO(&rset);
    int c = 0;
    for ( ; ; ) {
      c = c+1;
      if (stdineof == 0) {
          FD_SET(fileno(fp_in), &rset);
      }
      FD_SET(sockfd, &rset);
      maxfdp1 = max(fileno(fp_in), sockfd) + 1;
      select(maxfdp1, &rset, NULL, NULL, NULL);

      if (c == 20)
          break;

      if (FD_ISSET(sockfd, &rset)) {
          if (read(sockfd, response, MAXLINE) == 0) {
              if (stdineof == 1) {
                  return;
              } else {
                  printf("client: server terminated prematurely");
                  exit(0);
```

```
            }
        }
        fputs(response, stdout);
    }

    if (FD_ISSET(fileno(fp_in), &rset)) {
        if (fgets(message, MAXLINE, fp_in) == NULL) {
            stdineof = 1;
            shutdown(sockfd, 1);
            FD_CLR(fileno(fp_in), &rset);
            continue;
        }
        n = send(sockfd, message, strlen(message), 0);
    }
  }
}
...
```

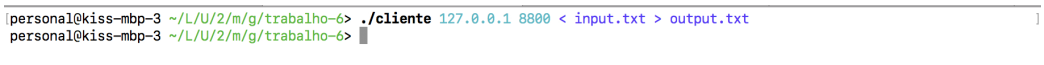
Já o program servidor sofreu pequenas alterações, pois agora o servidor não precisa executar os comandos enviados pelo cliente, mas sim retornar a mensagem novamente ao cliente. O seguinte trecho de código ilustra as alterações realizadas no programa servidor:

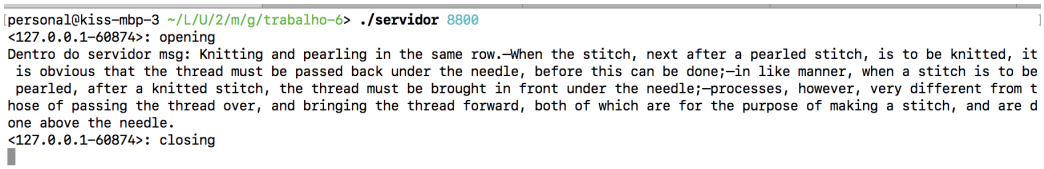
```
...
void doit(int connfd, struct sockaddr_in clientaddr) {
  char recvline[MAXDATASIZE + 1];
  int n;
  socklen_t remoteaddr_len = sizeof(clientaddr);

  while ((n = read(connfd, recvline, MAXDATASIZE)) > 0) {
    recvline[n] = 0;
    printf("Dentro do servidor msg: %s", recvline);
```

```
        send(connfd, recvline,  strlen(recvline),0);
    }
}
...
```

Para verificação, executamos o programa servidor por meio do comando `./servidor 8800` e realizamos uma conexão por meio do comando `./cliente 127.0.0.1 8800 < input.txt > output.txt`. Os resultados podem ser visualizados nas imagens a seguir:

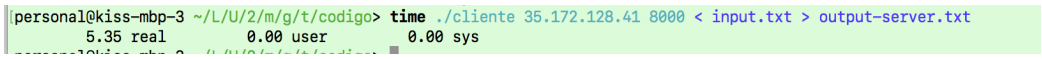




O conteúdo do arquivo de entrada `input.txt` mostrou-se igual ao arquivo retornado pelo servidor `output.txt`: *Knitting and pearling in the same row.—When the stitch, next after a pearled stitch, is to be knitted, it is obvious that the thread must be passed back under the needle, before this can be done;—in like manner, when a stitch is to be pearled, after a knitted stitch, the thread must be brought in front under the needle;—processes, however, very different from those of passing the thread over, and bringing the thread forward, both of which are for the purpose of making a stitch, and are done above the needle.*

2 Questão 2

No código implementado no passo 1, realizamos uma pequena alteração para que o programa cliente encerra-se a conexão após receber os dados do servidor. Sendo assim, por meio da inserção do comando `time`, calculamos o tempo total de execução do programa cliente para um grande arquivo de texto. Para validarmos de maneira mais precisa o desempenho de cada programa, utilizamos um sevidor remoto com `host` na Amazon AWS [2], configurado uma região distante em relação ao cliente a fim de obtermos uma latência na transmissão dos dados. A seguir, temos os resultados obtidos para o programa cliente atual (terminal em branco) e para o programa do exercício 5 (terminal em verde):



Nota-se uma diferença significativa entre a implementação realizada no exercício 5 e no exercício 6: a implementação utilizando-se `select` se mostrou 30 vezes mais rápida. Uma chamada de bloqueio para leitura provavelmente será a solução mais eficiente, pois chamadas não-bloqueantes (causado pelo uso da função `select`) são mais rápidas no contexto do manuseio de vários sockets em uma única thread, que é o caso deste exercício.

Bibliografia

[1] http://www.opengroup.org/onlinepubs/009695399/functions/select.html

[2] https://aws.amazon.com/pt/ec2/