**Trabalho Laboratorial nº1:**

Ligação de dados

**Turma 3 Grupo x:**

Antero Gandra up201607926

Francisco Moreira up201607929

Matheus Kozuki

**Sumário**

Realizado no contexto da unidade curricular de Redes de Computadores, foi requisitado aos alunos a implementação de um protocolo de comunicação assíncrona para a transmissão de um ficheiro através de uma porta série RS-232.

Na realização deste projeto tirámos várias conclusões sendo que as principais foram que (1) e que (2)

**Introdução**

**Objetivos do trabalho**

**Objetivos do relatório:**

O relatório emcontra-se dividido nas seguintes secções:

**Arquitetura –** onde se descrevem os diferentes blocos funcionais e interfaces;

**Estrutura do código –** onde se descrevem as APIs, as principais estruturas de dados, as principais funções e a relação das funções com a arquitetura); **Casos de uso principais -** onde se identificam os casos principais de uso e as sequências de chamada de funções para cada; **Protocolo de ligação lógica-** onde se indentificam os principais aspectos funcionais e se descreve a sua estratégia de implementação;

**Protocolo de aplicação -** onde se indentificam os principais aspectos funcionais e se descreve a sua estratégia de implementação; **Validação -** onde se descrevem os testes efectuados; **Eficiência do protocolo de ligação de dados-** onde fazemos uma caraterização estatística da eficiência do protocolo,com recurso a medidas sobre o código desenvolvido.  **Conclusões -** onde resumimos a informação apresentada nas secções anteriores e refletimos sobre os objectivos de aprendizagem alcançados

**Arquitetura**

**Camadas de protocolo**

O trabalho está organizado em duas camadas – *layers* – que permitem o correto funcionamento da aplicação: a camada do *protocolo de ligação de dados* e a camada de *aplicação*, que estão implementados em diferentes ficheiros *source* e *header*. Os ficheiros **link.c**e **link.h** representam a camada de ligação de dados enquanto os ficheiros **application.c** e **application.h** representam a camada de aplicação.

A camada do protocolo de ligação de dados contém funções genéricas que relacionadas com a porta série, tratando do estabelecimento da ligação e da transferência de dados e deteção de erros.

A camada de aplicação por outro lado contém funções mais especificas relacionadas, neste caso, com a transferência de um ficheiro pela porta serie.

**Interface e opções**

A interface é proporcionada pelos ficheiros da camada da aplicação que indica informação sobre o estado da transferência do ficheiro. No entanto, a camada de ligação de dados pode por vezes colocar informação na interface quando ocorrem erros.

As opções de ligação como *baud rate, timeout, timeout tries* e tamanho máximo da mensagem podem ser alteradas no ficheiro **settings.txt**. Adicionalmente existe uma opção neste ficheiro que indica a possibilidade de ser gerado um erro aleatorio em cada mensagem transferida. Esta opção foi usada para testar e comparar a eficiência do protocolo, deve obviamente ser colocada a 0 para uso real.

**Estrutura do código**

O código encontra-se dividido em 3 ficheiros correspondentes às 2 camadas da arquitetura e um outro ficheiro **alarme.c** que trata de processar timeouts gerados pelo sinal SIGALRM do sistema.

Naturalmente cada um destes ficheiros possui um header file correspondente onde estão declaradas as funções e estruturas de dados usadas, assim como algumas macros.

Ligação de dados (**link.c** e **link.h**):

* Funções:
  + llopen: estabelece a ligação entre o ‘writer’ e o ‘reader’.
  + llclose: fecha a ligação e entre o ‘writer’ e o ‘reader’.
  + llwrite: envia uma mensagem e espera pela resposta.
  + llread: lê uma mensagem e verifica os erros.
  + stuff: realiza o stuffing das mensagens.
  + destuff: realiza o destuffing das mensagens.
  + processBCC: calcula o BCC.
  + receiveMessage: lê uma mensagem. (é chamada pelo llwrite).
  + identifyMessageControl: identifica se uma mensagem tem o tipo previsto.
  + sendCommand: prepara e envia um comando.
  + sendMessage: prepara e envia uma mensagem.
  + printStats: imprime na linha de comandos as estatisticas da ligação.
  + statisticsSetup: inicializa as estatisticas da ligação.
  + connectionSettings: lê as settings pretendidas para a ligação.
* Estruturas de dados:
  + State: Estados da máquina de estados
  + Mode: Modo da aplicação, ‘writer’ ou ‘reader’
  + Control: Tipo do comando de controlo
  + MessageType: Tipo de mensagem
  + ErrorType: Tipo de erro
  + Data: lista de caracters e tamanho da informação, usado em Message
  + Message: Informação de uma mensagem
  + Stats: Estatisticas da ligação
  + Settings: Opções da ligação

Aplicação (**application.c** e **application.h**):

* Funções:
  + main: Inicio do programa e processamento dos parametros
  + sendFile: Envio de um ficheiro
  + receiveFile: Receção de um ficheiro
  + receiveControl: Receção de um pacote de controlo
  + receiveData: Receção de um pacote de dados
  + sendControl: Envio de um pacote de controlo
  + sendData: Envio de um pacote de dados
  + fileSize: Lê o tamanho de um ficheiro
  + openFile: Abre um ficheiro
* Estruturas de dados:
  + AppControlType: Tipo de pacote
  + Parameter: Tipo de parametro no pacote de controlo (Size ou Name)

Alarme (**alarm.c** e **alarm.h**):

* Funções:
  + alarmHandler: Handler do alarme.
  + setAlarm: Configura o alarme.
  + stopAlarm: Para o alarme.

TODO: Esta descriçao detalhada é na parte Protocolo de ligaçao lógica/ Protocolo de aplicação:

int llopen() : Abre a ligação pela porta série com as configurações escolhidas e estabelece a ligação enviando um comando SET do ‘writer’ e aguardando pela resposta UA do ‘reader’.

int llclose(int fd) : Estabelece o fim da ligação pela porta série trocando comandos DISC entre o ‘writer’ e o ‘reader’ e por fim um comando UA do ‘writer’ para confirmar o fim da ligação. Por fim as definições iniciais da porta série são repostas.