Trabalho 6 – parte 2

- Leitura do teclado
  - Aplicações que requerem entrada de dados através do teclado, via terminal serial, devem utilizar a chamada de sistema *read*
    - `int read (char *buffer, int size)`
      - *buffer*: endereço onde devem ser armazenados os caracteres lidos
      - *size*: número máximo de caracteres que podem ser armazenados em *buffer* contando o final de *string* 0
      - Retorna
        - 0 caso <ENTER> ainda não tenha sido pressionado ou
        - o número de caracteres armazenados após <ENTER> ter sido pressionado
      - *buffer* deve conter uma *string* finalizada com 0. <ENTER> não deve ser armazenado

Trabalho 6 – parte 2

- Leitura do teclado (tratamento no kernel)
  - O *handler* do módulo RX deve ler o caracter recebido pelo terminal, enviá-lo de volta via módulo TX e armazená-lo em uma *string* de tamanho fixo (e.g. 80 caracteres)
    - Esta *string* é uma variável do kernel
  - Ao detectar o código ASCII do <ENTER>, o *handler* deve finalizar a *string* com 0 ('\0')
  - Após a finalização da *string*, a chamada de sistema *read* está apta a copiar *size* caracteres da *string* do kernel para *buffer* e retornar para a aplicação o número de caracteres copiados (sincronização entre *handler* RX e *read*)
  - Após retornar o número de caracteres copiados em *buffer*, *read* deve voltar a retornar 0 até que <ENTER> seja pressionado novamente

Trabalho 6 – parte 2

- Leitura do teclado
  - Quando uma aplicação deseja ler dados do teclado, ela deve fazer *polling* na chamada de sistema *read* até que ela retorne um valor diferente de 0

```
...

/* Leitura bloqueante */
while (read(buffer,size) == 0);

...
```

Bloqueia até o *handler* RX detectar <ENTER>

Após o bloqueio, *buffer* contém a *string* digitada

Trabalho 6 – parte 2

- Aplicação
  - Inicialmente a aplicação deve ler uma *string* de no máximo 80 caracteres do teclado e mostrá-la invertida
  - Em seguida deve-se entrar em um *loop* infinito envolvendo o *bubble sort*, solicitando ao usuário
    1. Tamanho do *array* a ser ordenado
    2. Elementos do *array* (um-por-um)
    3. Tipo de ordenação (crescente/decrescente)
    4. Apresentação do *array* ordenado
    5. Volta ao passo 1