DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

# Algoritmos e Estruturas de Dados
## Aula prática #01

# Conteúdo

# 1 Objectivos

Pretende-se com esta aula familiarizar o aluno com metodologias de trabalho no desenvolvimento de código. São abordados os seguintes tópicos: desenvolvimento, compilação e depuração (debug) de programas em C usando a linha de comando (gcc e gdb); análise de memória alocada e libertada (valgrind).

No final da aula os alunos deverão:

- conhecer um conjunto mínimo de comandos UNIX, [3] – listar conteúdo da directoria, mudar de directoria, apagar ficheiros, mover ficheiros, criar e apagar directorias.
- saber como procurar ajuda na utilização de comandos UNIX.
- utilizar um editor de texto e conhecer um conjunto mínimo de operações de edição.
- reconhecer a utilidade do editor no desenvolvimento de código, com especial ênfase na indentação correcta, verificação de chavetas e parêntesis, etc..
- saber como compilar um programa usando o comando gcc – com opções de degub, definindo o nome do executável, etc..
- perceber a metodologia básica de depuração de erros de compilação, interpretando as mensagens de erro produzidas na compilação e procurando as suas causas.
- conhecer o ambiente de depuração em runtime gdb, [2], e um conjunto mínimo de comandos – listar linhas de código, criar e remover breakpoints, em linha e em função; executar passo a passo, com e sem entrada nas funções; examinar conteúdo de variáveis; usar o menu de ajuda do gdb.
- conhecer a metodologia básica de depuração em runtime.
- saber usar o comando valgrind na análise de requisitos de memória de um programa.

# 2 Plano de aula

Para atingir os objectivos anteriormente listados propõe-se o seguinte plano de aula.

1. Ilustrar a utilização de comandos UNIX e do comando man.
2. Ilustrar um episódio de edição de código com o emacs.
3. Abrir o ficheiro `class1a.c` com o emacs e produzir algumas alterações (como retirar `";"` ou retirar `"}"`) e ver o que acontece na tabulação.
4. Compilar o ficheiro `class1a.c` com o comando `gcc -Wall class1a.c -lm` e registar as linhas onde surgem mensagens de erro (para facilitar, talvez seja boa ideia ter duas sessões abertas: uma para a compilação e outra para a edição).

   Os alunos deverão tentar identificar os problemas e propor soluções.
5. Depois de depurado em compilação, mostrar o resultado de compilar com `gcc -Wall class1a.c -lm` ou com `gcc -Wall -o c1a class1a.c -lm`.

6. A primeira versão do programa vai crashar, pelo que se terá de produzir novo executável com opção de debug: `gcc -Wall -g -o c1a class1a.c -lm`.

7. Correr o gdb e ilustrar um conjunto básico de comandos: `help`, `list`, `break`, `next`, `step`, `continue`, `delete`, `print`, `backtrace`, etc..

8. Ilustrar o procedimento de depuração em runtime utilizando o gdb, corrigindo os erros identificados.

9. Depois de o programa correr sem problemas executar o comando:

   ```
   valgrind --leak-check=full ./c1a <argumento>
   ```

   Identificar e corrigir as causas das mensagens de erro produzidas.

# Referências


[1] GNU Emacs Reference Card.

[2] GDB Quick Reference.

[3] J. Sequeira, Guia de referência de comandos Unix, IST, 2000.