Guião 3

Objetivos

- Compreender o conceito de programa
  - conceito de programa (entrada, processamento e saída)
  - conceito de aplicação (na *stack* SO)
  - processos "pai" e "filho"
- Compreender que um ficheiro é composto por bits
  - código ASCII
  - código Assembly
  - ficheiro executável (*shell*)
- Compreender a estrutura de um progrma
  - variável e função
  - tipos int, char e float
- Compreender o procedimento de compilação
  - diretivas `-S` e `-g`
  - programas `gcc`, `xxd`, `objdump` e `gcc`

Para antes da aula

1. Assista o vídeo https://www.youtube.com/watch?v=RuKkePyo9zk e calcule a representação binária para 0.1 em IEEE 754.
2. Experimente o conversor: https://www.h-schmidt.net/FloatConverter/IEEE754.html
3. Consulte o livro recomendado para a UC de Sistemas de Computação, Essentials of Computing Systems, em especial as secções 3.5 e 3.6, pelo link: https://ebooks.uminho.pt/index.php/uminho/catalog/view/93/141/1786

O Código Fonte

```
#include <stdio.h>
int main(int argc, char **argv) {
  float a = 2.0, b = 2.75, c = -(a + b);

  float  i = 0.1 + 0.1;
  double j = 0.1 + 0.1;
  printf("i: %f\n", i);
  printf("j: %f\n", j);

  int variavel;
  variavel = argv[0][0];
  return variavel; // ATENÇÃO: isto não se faz!
}
```

1. Salvar o código acima num ficheiro `prog.c`
2. Compilar o código com `gcc prog.c`

3. Tipos primitivos
   - `int` números em binários e haxadecimal
   - `char` ver tabela ASCII (https://www.ascii-code.com/)
   - `float` padrão IEEE 754 (https://www.h-schmidt.net/FloatConverter/IEEE754.html)

O Ficheiro Binário (em ASCII)

Considere executar os comandos abaixo passando-os ao comando `less`. A título de exemplo, para o primeiro comando é preferível escrever

```
xxd -b prog.c | less
```

1. Use o comando `xxd -b prog.c` para ver o ficheiro em binário
2. Use o comando `xxd prog.c` para ver o ficheiro em hexadecimal
3. Com base na tabela ASCII (https://www.ascii-code.com/) interprete o output
4. Use o comando `cat a.out` para ver o conteúdo do ficheiro (em ASCII)
   - use o `cat -v a.out` para os caracteres não imprimíveis
   - use o `cat --help` e o `xxd --help` para outras opções
5. Use o comando `xxd a.out` para ver o ficheiro em hexadecimal
   - use o comando `xxd -i a.out` para ver o ficheiro no estilo C

Correndo o Código

1. Executar o binário com `./a.out 2`
2. Ver o output com o comando `echo $?`
   - o `argv` é um *array* de `char` logo,
     - argv[0] = "./a.out" e argv[1] = "2" == 50 (em ASCII decimal), assim
     - argv[0][0] = '.' == 46, argv[0][1] = '/' = 47, ...
3. Altere os parametros no programa e veja o que altera

O Ficheiro Assembly

1. Gere o *assembly* do código com `gcc -S prog.c` e seguidamente execute `cat prog.s`
2. Gere um binário para *debugging* com `gcc -g prog.c`
   - Use `objdump -S a.out` para ver o *assembly*
   - Relacione os endereços do `objdump` com o do `xxd` e observe como os mesmos bytes aparecem em ambos os ficheiros, num interpretado como ASCII e noutro como *assembly*.

Observar o *Run time* do Código

1. Gere um binário para *debugging* com `gcc -g prog.c`
2. Inicie o GDB com `gdb a.out`
3. Mostre o programa que foi carregado com `list`
4. Coloque um breakpoint na função `main` com `break main`
   - execute `run 2` e após parar no breakpoint, qual o valor do `argc` e do `argv`?
   - mostre o valor do `argc` com `p argc` (mostra o número de parâmetros)
   - qual o valor para `p argv[1][0]`?
   - use `continue` para executar o resto do programa
     - deve exibir uma mensagem como `[Inferior 1 (process xxxxx) exited with code 062]`, aqui o código está em ASCII octal.
5. Veja o efeito de diferentes modificadores de tipo
   - `p /t 2`
   - `p /t -2`
   - `p /t (short) -2`
   - `p /t (long) -2`
   - `p (unsigned) 2`
   - `p (unsigned) -2`
   - `p /t (unsigned) -2`
   - `p /t (unsigned) 2`
6. A representação dos inteiros chama-se *complemento para dois*
   - consulte a secção correspondente no livro de SC e represente os números -9 e 11
   - seguidamente imprima ambos os números com `p /t` para comparar os resultados com o que tinha calculado no ponto anterior

Sobre o tipo `float`

1. Corra o `prog` e veja os valores para `i,j`
2. Inicie o gdb com o `prog`
3. Avance até o `main` com `start`
4. Depois avance até a linha 8 com `step`
   - Mostre o valor de `i` com `p`. Compare com o resultado do `printf` e explique a razão disso.
   - Mostre o valor de `j` com `p`. Compare com o resultado do `i` e explique a razão disso.
   - Altere o valor de `i,j` para 0.3 com `set variable i = 0.3` e exiba ambos valores com `p`.
   - Altere o valor de `i,j` para 0.1+0.2 com `set variable` e exiba ambos valores com `p`.
   - Repita o procedimento no programa e veja como o `printf` se comporta.
5. Altere o `i` para 0.1 com `set variable`

- Veja a representação em binário com `x /t &i`
- Use o conversor (https://www.h-schmidt.net/FloatConverter/IEEE754.html)