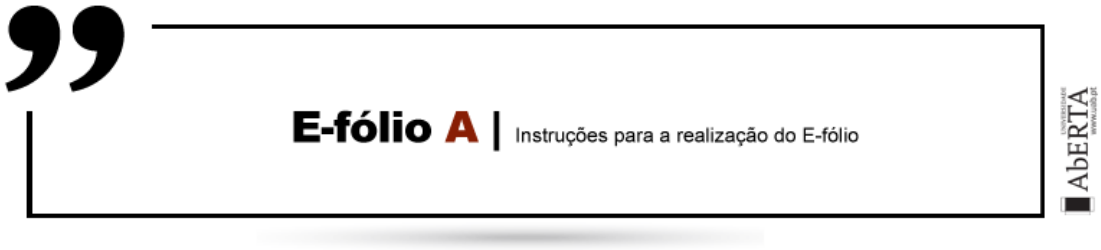
UNIVERSIDADE ABERTA www.uab.pt

Código 21111

Sistemas Operativos

(ano letivo 2021-22)



Este enunciado constitui o elemento de avaliação designado por "e-fólio A" no âmbito da avaliação contínua e tem a cotação total de 3 valores. A sua resolução deve ser entregue até às 23h55 do dia 11 de abril pelos alunos que escolheram a modalidade de avaliação contínua.

A resolução deve ser entregue através de um único ficheiro compactado .zip, que:

(i) contém os ficheiros .c que constituem o código dos programas, prontos a serem compilados;

(ii) contém um ficheiro de nome relatorio.pdf (sem acento) com um relatório simples e sucinto com informações solicitadas e/ou complementares de modo a permitir uma fácil compreensão do trabalho realizado. É desnecessário incluir uma listagem integral do código.

(iii) O nome do ficheiro .zip a entregar deve seguir a seguinte convenção para o seu nome,

"NumeroAluno-PrimeiroNome-Apelido-21111-efA.zip"

Por exemplo, um aluno com número 327555 e nome Paulo ... Costa, deverá dar o seguinte nome ao ficheiro, "327555-Paulo-Costa-21111-efA.zip", (sem acentos).

O ficheiro deve ser única e exclusivamente entregue através do recurso "E-fólio A" disponibilizado na plataforma (Nota: apenas é visível para os alunos inscritos em avaliação contínua), não sendo aceites trabalhos enviados por outras vias, como por exemplo por e-mail.

Esta é uma prova de avaliação **individual** e não "um trabalho de grupo". A sua resolução deve provir unicamente do conhecimento adquirido e trabalho original desenvolvido pelo próprio aluno. Os alunos deverão saber distinguir claramente entre discutir os conteúdos abordados na unidade curricular (permitido) e discutir a resolução específica do e-fólio (não permitido).

No caso de dúvidas de interpretação do enunciado, utilize o fórum de avaliação para pedidos de esclarecimento.

1. [3] Escreva um programa em linguagem C padrão, de nome `teste.c`, que permita efetuar o teste de um programa. O programa recebe dois argumentos, com o programa a testar e os casos de teste:

```
teste <programa.c> <casos.txt>
```

O `programa.c` recebe todos os dados de entrada pelos argumentos do programa, e escreve todo o output para o *stdout*.

O programa de teste deve efetuar as seguintes operações:

- **a)** compilar <`programa.c`> (caso exista erro, terminar)
- **b)** ler o ficheiro <`casos.txt`> e abrir um novo processo por cada caso de teste
- **c)** para cada processo de um dado caso de teste, o programa é executado e o seu output gravado num ficheiro `out<id>.txt`, sendo id um índice do caso de teste, iniciando em 0
- **d)** após todos os casos de teste terminarem, o processo principal compara os resultados com os corretos, assinalando os casos de teste falhados, e indicando o número de testes executados

O formato do ficheiro <`casos.txt`> é constituído por um número arbitrário de linhas, sendo as linhas ímpares os argumentos de entrada do programa, e as linhas pares imediatamente seguintes, a saída esperada do programa. Exemplo de conteúdo:

```
1
Erro: N tem de ser maior que R e este maior que 0.
2 2
1
```

Este ficheiro tem 4 linhas, tendo, portanto, 2 casos de teste. O primeiro caso de teste tem como entrada "1", e a saída deverá ser o texto da segunda linha. O segundo caso de teste tem como entrada "2 2" e a saída deverá ser "1".

Existem no máximo 20 casos de teste, e cada linha do ficheiro tem no máximo 255 caracteres.

Por cada processo distinto criado, deve nesse mesmo instante ser impresso uma mensagem do tipo "`Processo PID=xxx PPID=xxx`". Neste trabalho a criação de processos deve ser feita recorrendo unicamente à função de sistema **fork()**. Deve ser testado se ocorre erro na chamada à função de sistema **fork**(), caso em que o programa termina.

Não deve ser utilizada a função de biblioteca **system()**. Pode substituir a imagem de um processo com recurso a uma ou mais funções sistema diferentes da família **exec()** de entre o conjunto {**execl, execlp, execv, execvp**}.

São fornecidos em anexo, dois programas com dois ficheiros de casos de teste (produto.c, combinacoes.c, casosproduto.txt, casoscombinacoes.txt). Estes dois programas são exercícios das atividades formativas da UC de Introdução à Programação.

Nota: para redirecionar o output para um ficheiro (com o nome na string `str`), necessário no passo c), deve executar antes de chamar uma função **exec**, as seguintes instruções:

```
stdout = freopen(str, "wt", stdout);
```

Apresenta-se a seguir um exemplo de execução do programa:

```
$ ./teste produto.c casosproduto.txt
Processo PID = 4458 PPID = 252 [principal].
Processo PID = 4459 PPID = 4458 [compilacao].
Processo PID = 4464 PPID = 4458 [execucao caso 0].
Processo PID = 4465 PPID = 4458 [execucao caso 1].
Processo PID = 4466 PPID = 4458 [execucao caso 2].
Processo PID = 4467 PPID = 4458 [execucao caso 3].
Processo PID = 4468 PPID = 4458 [execucao caso 4].
Processo PID = 4469 PPID = 4458 [execucao caso 5].
Processo PID = 4470 PPID = 4458 [execucao caso 6].
Processo PID = 4471 PPID = 4458 [execucao caso 7].
Processo PID = 4472 PPID = 4458 [execucao caso 8].
Processo PID = 4473 PPID = 4458 [execucao caso 9].
Processo PID = 4474 PPID = 4458 [execucao caso 10].
Erro caso 6: esperado '5040999' obtido '5040'.
Casos de teste executados: 11.
```

- O programa deve estar identificado com um cabeçalho similar ao seguinte,

```
/*
** UC: 21111 - Sistemas Operativos
** e-fólio A 2021-22 (teste.c)
**
** Aluno: 327555 - Paulo Costa
*/
```

Critérios de correção:

- Programa desenvolvido difere significativamente das especificações e instruções do enunciado => 0 valores.
- Programa não compila ou produz avisos (warnnings) com `gcc -Wall` => 0 valores.
- Código do programa não está correta e uniformemente indentado de modo a permitir a sua leitura fácil => 0 valores
- Programa não está comentado => 0 valores. Os comentários no programa elucidam questões relevantes do código locais ao comentário.
- O programa em conjunto com o relatório não está estruturado, comentado ou explicado de modo à fácil compreensão da sua estrutura e funcionamento => 0 valores.
- O programa não funciona corretamente ou não cumpre todas as especificações ou é demasiado complexo => de 0 a 100% valores, sendo o programa avaliado como um todo e tendo em conta a implementação das características pedidas.

Nota ética: Nunca é de mais referir que o código a apresentar como solução para este e-fólio deve ser 100% original do aluno. A probabilidade de duas pessoas que efetivamente não comunicaram entre si, apresentarem programas "quase iguais" é considerada nula. Isto é válido para qualquer par de alunos (cópia), assim como entre um aluno e qualquer outra pessoa, em particular através da

Internet (cópia/plágio), onde existem inúmeras soluções e código para os mais variados problemas, em sites, fóruns, blogs, etc.
Cumpra estritamente as normas de realização individual, como se estivesse num exame com consulta, onde pode consultar a documentação mas não pode falar com ninguém.

Ajudas na resolução (para utilização apenas se estiver bloqueado):

- Para verificar se a compilação foi bem-sucedida, utilize wait(&status); e veja na documentação (man wait), como ver o valor retornado pelo compilador;
- Para executar um caso de teste, como não sabe quantos argumentos existem, utilize uma versão de exec com o argumento char *argv[]. Pode facilmente preencher argv com recurso à função strtok, utilizando como separador o espaço;
- Para garantir que os textos são colocados pela ordem esperada, efetuar todas as impressões do processo principal.
- Atendendo a que o **fork** duplica o processo, constitui também um fator de aumento de complexidade do código, superior a condicionais e ciclos. Sugere-se assim que seja utilizado o **fork** apenas em funções de reduzida dimensão, e não mais que um **fork** numa mesma função.

FIM