21111 - Sistemas Operativos: proposta de resolução do eFólio A

E-fólio A, notas lançadas
por José Coelho - Terça, 19 Abril 2022, 10:52

Caros estudantes,

Estão lançadas as notas do e-fólio A, podem ir ver a nota bem como alguns comentários. Vejam também a resolução proposta em anexo, e coloquem qualquer questão que tenham, aproveitando também o momento de lançamento de notas, para esclarecer algum dilema que tenham tido durante o e-fólio.

Em termos globais, tivemos dos 263 estudantes em avaliação contínua, apenas 70 entregas de trabalhos. Temos portanto uma baixa taxa de avaliação, embora seja comum no primeiro ano, cada estudante que se inscreve numa unidade curricular, deve reservar tempo para poder realizar as atividades formativas e as atividades de avaliação. Caso não tenha disponibilidade, deve procurar reduzir o número de unidades curriculares em que se inscreve, de modo a ter uma disponibilidade compatível com o trabalho necessário realizar em cada unidade curricular.

As notas dos estudantes que entregaram, são bastante boas, tendo ocorrido a seguinte distribuição dos estudantes pelas notas:

- [2,5; 3] valores: 27 estudantes
- [2; 2,5[valores: 22 estudantes
- [1,5; 2[valores: 5 estudantes
- [1; 1,5[valores: 4 estudantes
- [0.5; 1[valores: 11 estudantes
- [0; 0,5[valores: 1 estudante

A maior parte dos estudantes teve a nota máxima ou muito perto disso. Quem teve muitas dificuldades neste e-fólio, ou não chegou a entregar, deve procurar identificar os principais problemas que impediram de realizar este e-fólio em condições e procurar resolver, se necessário algum tipo de orientação podem expor o seu caso via fórum. Realço também o caminho que é possível, para quem não tenha entregue o e-fólio A ou tenha nota muito baixa, que é a possibilidade de no e-fólio B, atendendo a que vale 5 valores, recuperarem o que não conseguiram fazer neste e-fólio. Outra alternativa é a via do exame na época de recurso, mas não esquecer que no exame irão estar perguntas relacionadas com a matéria coberta nos e-fólios.

Esta semana inicia-se o módulo 3, "Gestão de Memória + Introdução à programação Multitarefa em Linux", tendo agora outro laboratório e sendo a matéria base para o e-fólio B. Após verem a nota e feedback individual, é tempo para iniciar os trabalhos.

Bom trabalho.

Cumprimentos,
José Coelho

Anexos: teste.c, casoscombinacoes.txt e casosproduto.txt

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>

#define MAX_STR 255
#define MAX_CASES 20

int CompilarPrograma(char* programa)
{
    int status;
    pid_t pid;

    if ((pid = fork()) == -1)
        exit(1);

    if (pid) { // processo pai
        // output do lado do processo principal, para evitar problemas de ordem das
linhas de output
        printf("\nProcesso PID = %d PPID = %d [compilacao].", (int)pid,
(int)getpid());
        fflush(stdout);
        wait(&status);
        if (WIFEXITED(status) && WEXITSTATUS(status) == 0)
            // compilation ok
            return 1;
    }
    else // processo filho
        execlp("gcc", "gcc", "-Wall", "-o", "tst", programa, NULL);
    return 0;
}

void LimparFimDeLinha(char* str) {
    while(strlen(str)>0 && strchr("\n\r", str[strlen(str) - 1])!=NULL)
        str[strlen(str) - 1] = 0;
}

int LerCasos(char* casos,
    char input[MAX_CASES][MAX_STR],
    char output[MAX_CASES][MAX_STR],
    int *ncasos)
{
    char str[MAX_STR];
    FILE* f;
    *ncasos = 0;
    if ((f = fopen(casos, "rt")) != NULL) {
        while (!feof(f)) {
            if (fgets(str, MAX_STR, f) != NULL) {
                strcpy(input[*ncasos], str);
                if (fgets(str, MAX_STR, f) != NULL) {
```

```
                    LimparFimDeLinha(str);
                    strcpy(output[(*ncasos)++], str);
                }
            }
        }
        fclose(f);
    }
    return *ncasos>0;
}

void ExecutarCaso(int i, char* arg)
{
    char *argv[10];
    char* pt;
    char str[MAX_STR];
    int j;
    pid_t pid;

    if ((pid = fork()) == -1)
        exit(1);

    if (pid==0) { // processo filho
        // preparar os argumentos
        argv[0] = "tst";
        j = 1;
        pt = strtok(arg, " ");
        while (pt != NULL) {
            argv[j++] = pt;
            pt = strtok(NULL, " ");
        }
        argv[j] = NULL;

        // redirecionar o output
        sprintf(str, "out%d.txt", i);
        stdout = freopen(str, "wt", stdout);

        execvp("./tst", argv);
    }

    // processo pai sai de imediato, lan◆ando outro caso de teste de seguida
    // colocar o output do lado do processo principal, para evitar problemas de ordem
das linhas de output
    printf("\nProcesso PID = %d PPID = %d [execucao caso %d].", (int)pid,
(int)getpid(), i);
    fflush(stdout);
}

void VerificarCaso(int i, char* output)
{
    char str[MAX_STR];
    FILE* f;
    sprintf(str, "out%d.txt", i);
```

```
    if ((f = fopen(str, "rt")) != NULL) {
        strcpy(str, "");
        fgets(str, MAX_STR, f);
        LimparFimDeLinha(str);
        if (strstr(str, output) == NULL)
            printf("\nErro caso %d: esperado '%s' obtido '%s'.", i, output, str);
    }
    else
        printf("\nErro caso %d nao executado.", i);
}


int main(int argc, char **argv)
{
    char input[MAX_CASES][MAX_STR];
    char output[MAX_CASES][MAX_STR];
    int ncasos, i, status;

    if (argc != 3) {
        printf("\nUtilizacao: teste <programa.c> <casos.txt>");
        exit(0);
    }

    printf("\nProcesso PID = %d PPID = %d [principal].", (int)getpid(),
(int)getppid());

    // 1. compilar o programa
    if (CompilarPrograma(argv[1])) {
        // 2. ler o ficheiro casos.txt
        if (LerCasos(argv[2], input, output, &ncasos)) {
            // 3. executar os casos de teste, um processo por cada caso de teste
            for (i = 0; i < ncasos; i++)
                ExecutarCaso(i, input[i]);

            // 4. aguardar que todos os processos terminem
            for (i = 0; i < ncasos; i++)
                wait(&status);

            // 5. processar resultados
            for (i = 0; i < ncasos; i++)
                VerificarCaso(i, output[i]);

            printf("\nCasos de teste executados: %d.\n", ncasos);
        }
        else
            printf("\nErro ao ler os casos de teste.\n");
    }
    else
        printf("\nErro na compilacao.\n");

    return 0;
}
```

casoscombinacoes.txt (12 casos de teste: linha ímpar < input, linha par > output)

1
Erro: N tem de ser maior que R e este maior que 0.
2 2
1
3 2
3
5 2
10
15 2
105
15 5
303
22 5
26334
22 15
170544
220 150
-13145360
220 3
1750540
220 2
24090999
21 12
293930

casosproduto.txt (11 casos de teste: linha ímpar < input, linha par > output)

1
1
2
2
3
6
4
24
5
120
6
720
7
5040999
8
40320
9
362880
10
3628800
11
39916800