

”

E-fólio A | Folha de resolução para E-fólio



UNIDADE CURRICULAR: Sistemas Operativos

CÓDIGO: 21111

DOCENTE:

A preencher pelo estudante

NOME: Hugo José Costa Correia

N.º DE ESTUDANTE: 2405276

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 01/04/2025

TRABALHO / RESOLUÇÃO:

Foi fornecido um template “avp.c” pelo docente, e tendo um número de aluno par, foi desenvolvida a função “void prob0()” cujo objetivo é criar 3 processos B,C,D, descendentes do próprio programa (A), e um processo E descendente do processo C. Não houve mais alterações ao template fornecido além desta função e da informação de aluno nos locais adequados.

Na função “void prob0()” desenvolvida, optei por utilizar um loop “for”, controlado pelo número de caracteres da variável “char *processos="ABCD"”. Ainda que no trabalho sejam pedidos apenas 3 processos, a opção por utilizar um loop permite que o programa seja facilmente escalável, bastando adicionar mais letras à variável, e evita também algumas repetições de comandos.

O programa começa por mostrar no ecrã “Problema 0”, mudar de linha, e mostrar o PID e PPID do processo do próprio programa (Processo A).

De seguida inicia o loop for de i=1 até ao tamanho da string processos-1 (i<strlen(processos)). Em cada loop, primeiro é executada a limpeza do buffer com fflush(stdout), e de seguida é criado um processo com a função fork(). Se o resultado do fork for -1, significa que houve um erro e aborta (exit(1)). Se o resultado do fork for 0 (valor de sucesso retornado dentro do child criado) mostra o carater da string “processos” correspondente ao índice i do loop atual (o índice das string começam no zero, portanto o i=1 corresponde à letra B) e os respetivos PID e PPID. Quando o carater da string no i atual for ‘C’, tal como solicitado no enunciado, é criando mais um descendente novamente com a função fork(), onde é efetuada a limpeza do buffer, e validado mais uma vez o sucesso, mostrando o processo e respetivos PID e PPID.

Para terminar os processos quer no loop for, quer no if do processo E, é utilizada a função exit(0).

Caso o resultado dos fork seja diferente de 0 e -1, significa que estamos no processo pai e é invocado um wait(NULL) para garantir que a execução só continua depois dos processos descendentes terem terminado.