//Eric Goulart - 2110878
//João Pedro Biscaia Fernandes - 2110361

Questão 1) Objetivo: Elaborar programa para criar 2 processos hierárquicos (pai e filho)...

R: Inicializamos int **n** e int **i** fora do fork para a inicialização valer para os 2 programas, apesar disso observamos que o valor de n é diferente, isso acontece porque no fork criamos 2 programas diferentes, que ficam em áreas de memória diferentes, com isso, duas variáveis n são criadas, para cada programa. Assim, ao printar as variáveis, a saída será diferente.
O comando waitpid faz o pai esperar o filho terminar para então rodar.

Questão 2) Objetivo: – Programar funcionalidades dos utilitários do unix - “echo”

R: argc é o número de argumentos passados para o programa e argv[] é um array de strings contendo os argumentos. O código itera sobre esses argumentos imprimindo-os na saída.

Questão 3) Objetivo: – Programar funcionalidades do utilitário do unix “cat”

R: Também utilizamos argc e argv mas dessa vez argv é um array de arquivos, utilizamos o comando getc para extrair o texto desse arquivo e printamos na saída, além disso copiamos tudo que tem no arquivo passado como parâmetro e escrevemos em outro arquivo. O comando aceita "-" para copiar/criar arquivos então para criar um arquivo temos que rodar ./meucat - exemplo.txt e pra copiar um arquivo no outro temos que fazer ./meucat arq1.txt - arq2.txt.

Questão 4) Objetivo - Programar uma shell e executar os seus programas meuecho, meucat e os utilitários do Unix: echo, cat, ls

R: Começamos utilizando o type_prompt só para aparecer o $ na tela igual na shell.

Criamos 3 funções:

read_command lê o que digitamos na shell e armazena em um buffer.
split_commands verifica o que armazenamos no buffer e separa por ‘ ’ o comandos e os parâmetros.
run_command nessa função reescrevemos o que já tínhamos feito nos outros exercícios meuecho e meucat para o programa saber o que fazer quando escrevermos esses comandos na shell.

na main nós chamamos as funções descritas acima e utilizamos o waitpid para garantir que um programa espere o outro terminar para começar.