POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Projeto nanoShell

"Alexandre Jorge Casaleiro dos Santos (2181593) e por André Luís Gil De Azevedo (2182634) declaram sob compromisso de honra que o presente trabalho (código, relatórios e afins) foi integralmente realizado por nós, sendo que as contribuições externas se encontram claramente e inequivocamente identificadas no próprio código. Mais se declara que os estudantes acima identificados não disponibilizaram o código ou partes dele a terceiros.".

2181593 – Alexandre Jorge Casaleiro dos Santos

2182634 - André Luís Gil De Azevedo

Trabalho de Projeto da unidade curricular de Programação Avançada

Leiria, novembro de 2020

1. Funcionalidades

Algumas considerações gerais:

Foram definidas várias constantes para as mensagens de erro de forma a deixar o código mais legível. A execução base do programa baseia-se na função *main* que após várias validações para validar se o programa foi iniciado com alguma opção especial, chama a função *nano_loop*.

Nesta função é introduzida a interface base da *nanoShell*, ficando o programa a aguardar que o utilizador insira comandos. Ao receber um comando toda a *string* é validada para garantir que não inclui algum caracter não suportado, se inclui o comando *bye* (que dá origem a terminar a *nanoShell* nesse momento), ou se é uma função de redireccionamento (padrão ou erro).

Caso tudo tenha corrido bem até aqui, é criado um processo filho recorrendo à função *fork* que irá redirecionar ou executar o comando com recurso à função *execvp* e termina de seguida. Após a criação do processo filho, no código principal (pai) esperamos por um sinal de que o filho executou com sucesso e terminou (através da função *wait*) para depois libertar memória. De seguida voltamos a aguardar pelo utilizador para introduzir um novo comando e continua até encontrar alguma condição de paragem ou algum erro que leve a terminar o programa.

1.1. Redireccionamento dos canais de saída e de erro padrão

Estado de funcionamento: *Totalmente operacional*

Implementação:

Conforme descrito nas considerações gerais, a *nanoShell* ao receber um comando, seja inserido pelo utilizador ou através da leitura de comandos de um ficheiro, valida toda a *string* de dados introduzidos. Neste processo de validação, após verificar que todos os caracteres introduzidos são válidos (função *nano_verify_char*), a *string* é dividida em "*tokens*" (separados por espaço) de forma a verificar a existência do sinal de redireccionamento em um desses *tokens*.

A função que faz a divisão em *tokens* é a *nano_split_lineptr* e a função que trata o redireccionamento é a *nano_verify_redirect* que devolverá um valor entre 1 a 4 em caso de sucesso, ou -1 em caso de não ter encontrado nenhum dos *tokens*.

Em caso de sucesso devolve o ficheiro para onde será redirecionado o comando de *output* (por ponteiro) e incrementa os contadores definidos numa estrutura de contadores global.

Na função *nano_exec_commands* onde são chamadas as funções descritas anteriormente, guarda o resultado da verificação e, após criação de um processo filho é então aberto o ficheiro para onde será redirecionada a execução do comando.

1.2. Captura de *Signals*

Estado de funcionamento: *Totalmente operacional*

Implementação:

A nanoShell por defeito captura os sinais SIGINT, SIGUSR1 e SIGUSR2, conforme os requisitos pedidos no enunciado do projeto. Para elaboração destas funcionalidades, existe uma definição da *sigaction act* no código da função *main* onde é passada a rotina de tratamento de sinais para a função *nano_sig_handler*.

Nesta função, como boa prática, guardamos a variável *errno* numa variável auxiliar para no final voltar a repor a variável. Além disso apenas é executado o código necessário para implementação das funcionalidades pedidas, em suma: no caso do SIGINT e do SIGUSR1 é impressa uma mensagem no *stdout*, no caso do SIGUSR2 é criado um ficheiro no diretório corrente com informações dos diversos contadores (comandos ou redireccionamentos).

1.3. Parâmetros da linha de comando - Help

Estado de funcionamento: *Totalmente operacional*

Implementação:

Se a *nanoShell* for iniciada com o parâmetro -h apenas irá imprimir um conjunto de informação que pretende ajudar o utilizador a utilizá-la, e irá imediatamente terminar o programa em execução.

Para esta funcionalidade não foram utilizadas funções ou estruturas de dados extra, sendo o código executado logo no início da função *main* após validação do *parsing* dos argumentos.

1.4.Parâmetros da linha de comando – Max

Estado de funcionamento: *Totalmente operacional*

Implementação:

Se a *nanoShell* for iniciada com o parâmetro -m irá iniciar um contador de comandos que irá terminar a sua execução quando o valor indicado a seguir ao parâmetro for atingido.

Esta funcionalidade está maioritariamente na função *main*, e começa por verificar se o valor introduzido é maior que 0, caso contrário imprime uma mensagem de erro e termina o programa. No caso de o valor ser válido então recorremos a uma variável global *counters* definida numa estrutura *NanoCounters* em que é colocado o valor do máximo de comados a executar. Ainda dentro da função *main*, mesmo no fim, caso a execução tenha decorrido como planeado e a funcionalidade ativada, será impresso o número de comandos executado.

O restante código desta funcionalidade, está no final da função *nano_loop* (chamada dentro da função *main*) onde verifica se já atingimos o máximo de comandos, e dentro da função *nano_verify_redirect* (que é chamada dentro da *nano_exec_commands*) onde é incrementado o contador de comandos depois de efetuadas várias validações para garantir que o comando é válido e que será executado.

1.5.Parâmetros da linha de comando – File

Estado de funcionamento: *Totalmente operacional*

Implementação:

Se a *nanoShell* for iniciada com o parâmetro -f irá ler e executar todas as linhas de comandos válidas que estão dentro do ficheiro passado por argumento, terminando o programa após concluir a leitura de todas as linhas do ficheiro.

Esta funcionalidade começa na função *main*, abrindo o ficheiro que vem no argumento, e caso a abertura do ficheiro tenha sucesso inicia um ciclo para ler as linhas com recurso à função *getline*. Caso a linha inicie com **#**, **\n** (*line feed*), **espaço** ou ***tab***, essa linha é automaticamente ignorada.

Caso contrário é removido o caracter **\n** da linha lida (para tornar o comando válido) e impresso o comando no *stdout* (para ajudar a perceber o que está a acontecer no ficheiro) antes de chamar a função nano_exec_*commands* que irá executar o comando após efetuar as validações necessárias.

Antes de terminar o programa, é libertada a memória de leitura das linhas do ficheiro, que é de imediato fechado.

1.6.Parâmetros da linha de comando – *Signal* File

Estado de funcionamento: *Totalmente operacional*

Implementação:

Se a *nanoShell* for iniciada com o parâmetro -s irá criar um ficheiro "signals.txt" na localização do programa, com os comandos que irão permitir enviar sinais para a nanoShell e que serão tratados. Esses comandos podem ser utilizados com a funcionalidade de captura de sinais descrita no tópico 1.3 deste relatório.