

Sistemas Operacionais II

Aula 3 - BASH e Redirecionamento de entrada/saída

Autor: Renê de Souza Pinto

Orientação: Prof. Dr. Francisco José Monaco

`rene@grad.icmc.usp.br`, `monaco@icmc.usp.br`

Universidade de São Paulo

Instituto de Ciências Matemáticas e Computação - ICMC

Escola de Engenharia de São Carlos - EESC

Sumario

- Introdução ao BASH
- Redirecionamentos
- Implementação do Redirecionamento
- Prática 3

Bash

- O BASH é o shell padrão na grande maioria das distribuições Linux
- Possui bastante recursos
- Robusto

Bash - Teclas de atalho

Podemos poupar horas de trabalho e digitação através de atalhos de teclado:

Ctrl + a	Vai para o começo da linha (mesmo que Home)
Ctrl + e	Vai para o final da linha (mesmo que End)
Ctrl + l	Limpa a tela
Ctrl + u	Limpa conteúdo da linha até a posição cursor
Ctrl + k	Limpa conteúdo da linha depois da posição do cursor
Ctrl + w	Apaga a ultima palavra
Ctrl + r	Busca reversa
Ctrl + t	Inverte os dois ultimos caracteres antes do cursor
Setas ↑ e ↓	Acessa histórico de comandos

Bash - Teclas de atalho

Podemos poupar horas de trabalho e digitação através de atalhos de teclado:

Alt + f	Avança para próxima palavra da linha
Alt + b	Volta para a palavra anterior da linha
Esc + t	Troca as duas ultimas palavras antes do cursor
Tab	Auto-completa um comando
Ctrl + c	Envia um sinal de interrupção para o processo em execução
Ctrl + z	Suspende o processo em execução
Ctrl + d	Sai do shell atual

Bash - Jobs

- Quando um processo é iniciado o BASH o inicia em foreground, ou seja, o terminal é travado até que o programa seja finalizado (ou interrompido com Ctrl+c, Ctrl+z, etc).
- Ctrl+z interrompe o programa, para retornar a execução:
 - fg n - Retorna em foreground (travando o terminal)
 - bg n - Retorna em background (deixando o terminal disponível)
 - n - Número do Job
- Para exibir os trabalhos do usuário:
jobs

Bash - Jobs

Exemplo:

```
$ find /usr > /dev/null
(Ctrl+z)
[1]+ Stopped                  find /usr > /dev/null
$ jobs
[1]+ Stopped                  find /usr > /dev/null
$ bg
[1]+ find /usr > /dev/null &
$ jobs
[1]+ Running                  find /usr > /dev/null &
```



Redirecionamento

Redirecionamento

- Recurso extremamente útil
- O S.O. possui 3 descritores de arquivos padrão:
 - 0: corresponde a entrada padrão (teclado, por exemplo)
 - 1: corresponde a saída padrão (monitor, por exemplo)
 - 2: corresponde a saída de erros padrão (monitor ou arquivo de log, por exemplo)
- Podemos redirecionar estas saídas

Redirecionamento

Redirecionamento de saída

>	Redireciona a saída de um comando para um arquivo especificado, inicializando-o caso não exista ou destruindo seu conteúdo anterior.
>>	Redireciona a saída de um comando para um arquivo especificado, anexando-o ao seu fim. Caso este arquivo não exista, será criado.
2 >	Redireciona os erros gerados por um comando para o arquivo especificado. Mesmo que não ocorra erro na execução do comando, o arquivo será criado.

Fonte: [1]

Redirecionamento

Redirecionamento de entrada

<	Avisa ao Shell que a entrada padrão não será o telado, mas sim o arquivo especificado.
<<	Também chamado de here document. Serve para indicar ao Shell que o corpo de um comando começa na linha seguinte e termina quando encontra uma linha cujo conteúdo seja unicamente o label que segue o sinal <<.

Fonte: [1]

Redirecionamento

Exemplos:

```
$ cat /proc/cpuinfo > my_cpu
$ cat /proc/devices >> my_cpu
$ cat /proc/cpuinfo | tee my_cpu2
$
$ cat > poema << FIM
O Shell é legal!
O Shell é maneiro!
Meu amigo companheiro,
Sou teu bom velho shelleiro,
Nunca me deixas em devaneio!
FIM
$
```

Pipes

- Saída de um lado é a entrada do outro
- Representado por |

Pipes

- Saída de um lado é a entrada do outro
- Representado por |

Exemplo:

```
$ ls -l | wc  
$
```

Implementacao

- Implementação do redirecionamento:
 - Abrir o arquivo de entrada/saída com as respectivas permissões
 - Fechar o descritor que será redirecionado (0, 1 ou 2)
 - Utilizar a *syscall dup()* para duplicar o descritor do arquivo aberto no local do descritor fechado

dup

- `int dup(int oldfd);`

Função: Duplica o descritor de arquivo *oldfd*, criando a cópia na menor posição possível na tabela de descritores

Entrada:

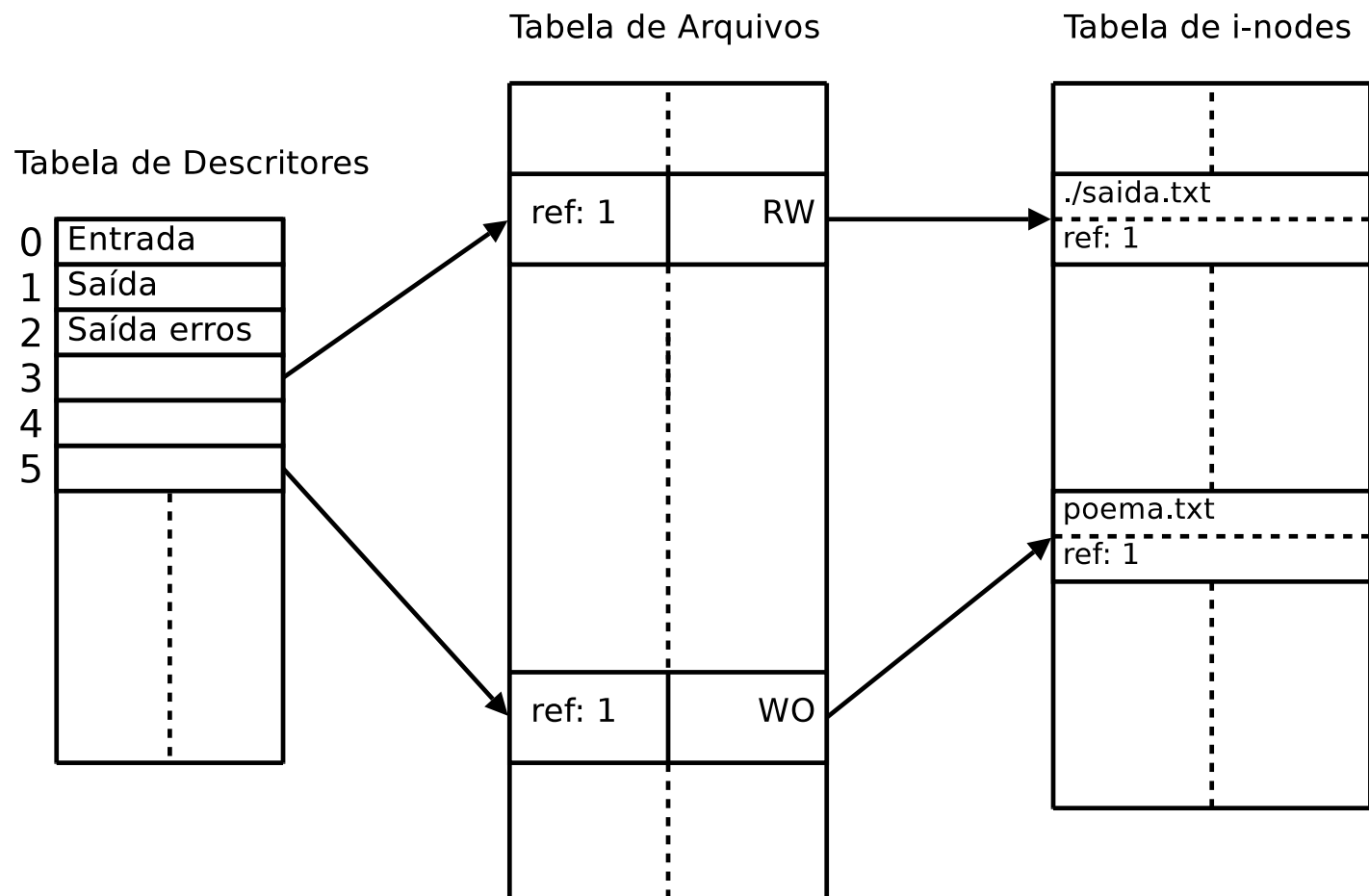
- `oldfd`: Descritor do arquivo aberto a ser duplicado

Retorno:

- `int`: Novo descritor (cópia) ou -1 se houve erro.

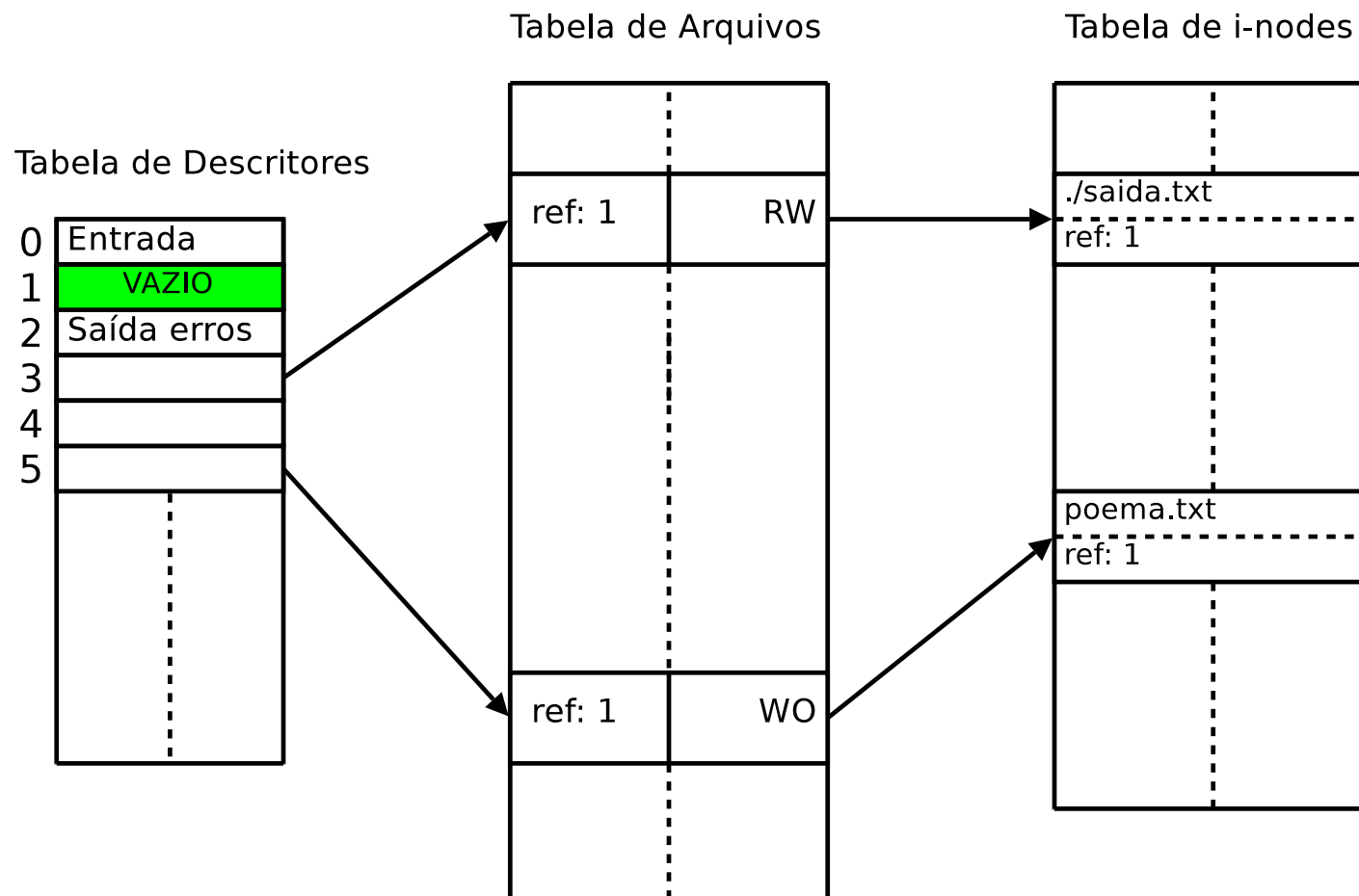
dup - ilustracao

- Tabela normal de descritores



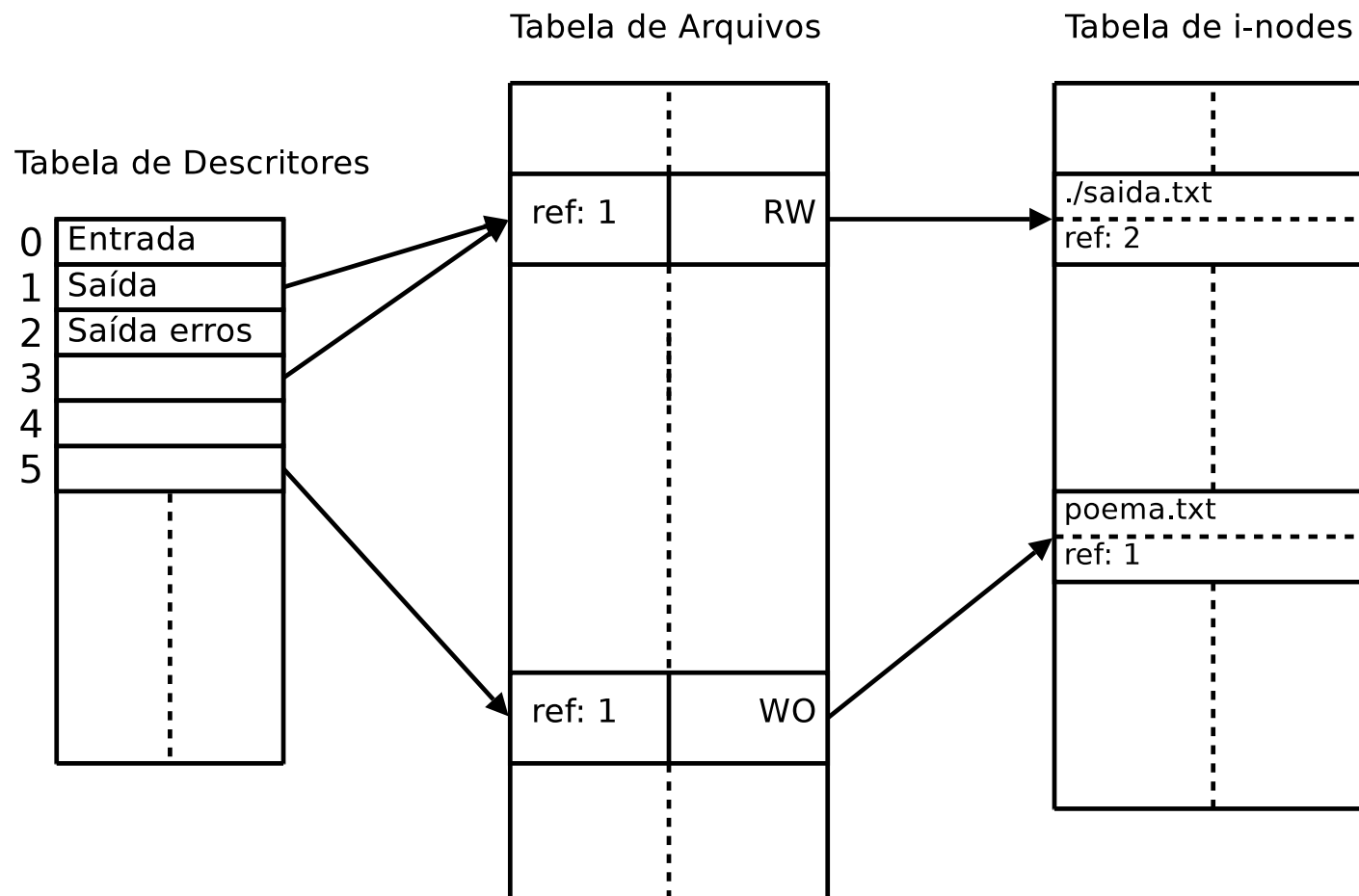
dup - ilustracao

- Saída padrão é fechada (`close(1);`)



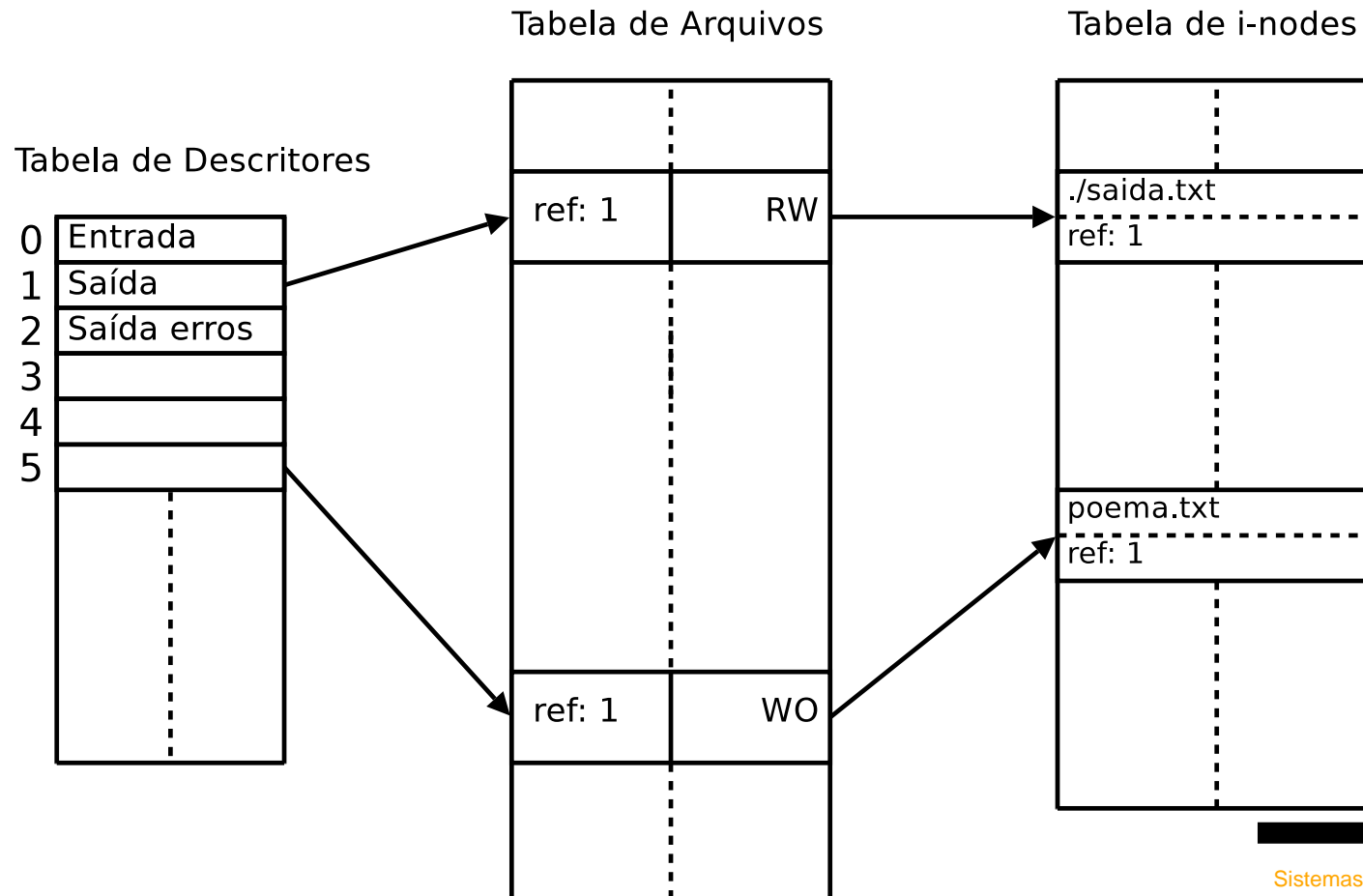
dup - ilustracao

- `dup(3)` é executado



dup - ilustracao

- O descritor 3 agora possui uma cópia em 1 (saída padrão), portanto pode ser fechado (`close(3);`)



Codigo exemplo

```
/* ... */

/* fork */
if( (child = fork()) ) {
    waitpid(child, &status, 0);
} else {
    if( (fd=open("./saida.txt",
                O_CREAT | O_TRUNC | O_WRONLY, S_IWUSR | S_IRUSR)) < 0)
        perror("Erro ao abrir/criar arquivo saida.txt\n");
    } else {
        /* Redireciona a saída */
        close(STDOUT_FILENO); /* STDOUT_FILENO = 1 */
        dup(fd);
        close(fd);
    }
    execve(line, NULL, environ);
}

/* ... */
```

Pratica 3

- Objetivo:
 - Familiarizar o aluno com os redirecionamentos e implementações para os mesmos.

Pratica 3

- Objetivo:
 - Familiarizar o aluno com os redirecionamentos e implementações para os mesmos.
- Faça um pequeno shell que suporte redirecionamento de entrada e saída. Os arquivos de entrada/saída podem ter nomes fixos, porém, implementar um *parser* que permita o usuário fornecer o nome do arquivo (ex: *comando > arquivo*) é extremamente interessante e poupará trabalho posterior.



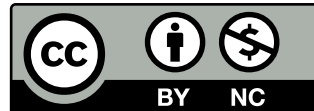
Bom Trabalho!

Bibliografia

Referências

- [1] Neves, Julio Cezar. Programação SHELL LINUX - 6^a edição. Brasport, 2006.
- [2] Linux manual pages
- [3] Renê de Souza Pinto, “Programação Shell Script: como dominar seu terminal”,
<http://renesp.com.br/materiais>

Licença



Este documento é licenciado sob a
Creative Commons Atribuição-Usó Não-Comercial 2.5
Brasil License.