

BSHELL

Gerado por Doxygen 1.7.4

Quinta, 17 de Novembro de 2011 05:17:28

Sumário

1	Índice dos Componentes	1
1.1	Lista de Componentes	1
2	Índice dos Arquivos	3
2.1	Lista de Arquivos	3
3	Classes	5
3.1	Referência da Estrutura cmd_info	5
3.1.1	Atributos	5
3.1.1.1	cmd_name	5
3.1.1.2	fun	5
3.2	Referência da Estrutura ENV_HISTORY	5
3.2.1	Atributos	5
3.2.1.1	end	5
3.2.1.2	his_cmd	6
3.2.1.3	start	6
3.3	Referência da Estrutura NODE	6
3.3.1	Atributos	6
3.3.1.1	cmd	6
3.3.1.2	next	6
3.3.1.3	num	6
3.3.1.4	pid	6
3.3.1.5	state	6
4	Arquivos	7
4.1	Referência do Arquivo Downloads/myshell/en_fou.c	7

4.1.1	Funções	7
4.1.1.1	init_environ	7
4.1.1.2	is_founded	8
4.1.2	Variáveis	8
4.1.2.1	cur_cmd	8
4.1.2.2	env_argv	8
4.1.2.3	env_buf	8
4.2	Referência do Arquivo Downloads/myshell/en_fou.h	8
4.2.1	Funções	8
4.2.1.1	init_environ	8
4.2.1.2	is_founded	8
4.3	Referência do Arquivo Downloads/myshell/jobs.c	9
4.3.1	Funções	10
4.3.1.1	add_job_node	10
4.3.1.2	bg_fun	10
4.3.1.3	ctrl_c	10
4.3.1.4	ctrl_z	10
4.3.1.5	del_node	10
4.3.1.6	fg_fun	10
4.3.1.7	find_node	10
4.3.1.8	jobs_fun	11
4.3.1.9	sig_init	11
4.3.2	Variáveis	11
4.3.2.1	end	11
4.3.2.2	fg_pid	11
4.3.2.3	head	11
4.3.2.4	p	11
4.3.2.5	q	11
4.4	Referência do Arquivo Downloads/myshell/jobs.h	11
4.4.1	Definições dos tipos	12
4.4.1.1	NODE	12
4.4.2	Funções	12
4.4.2.1	add_job_node	12
4.4.2.2	bg_fun	12

4.4.2.3	del_node	12
4.4.2.4	fg_fun	12
4.4.2.5	jobs_fun	13
4.4.2.6	sig_init	13
4.4.3	Variáveis	13
4.4.3.1	bg_pid	13
4.4.3.2	fg_pid	13
4.5	Referência do Arquivo Downloads/myshell/keypress.c	13
4.5.1	Funções	13
4.5.1.1	reset_keypress	13
4.5.1.2	set_keypress	13
4.5.2	Variáveis	14
4.5.2.1	stored_settings	14
4.6	Referência do Arquivo Downloads/myshell/keypress.h	14
4.6.1	Funções	14
4.6.1.1	reset_keypress	14
4.6.1.2	set_keypress	14
4.7	Referência do Arquivo Downloads/myshell/main.c	14
4.7.1	Funções	15
4.7.1.1	main	15
4.8	Referência do Arquivo Downloads/myshell/my_pipe.c	16
4.8.1	Funções	16
4.8.1.1	my_pipe	16
4.9	Referência do Arquivo Downloads/myshell/my_pipe.h	16
4.9.1	Funções	17
4.9.1.1	my_pipe	17
4.10	Referência do Arquivo Downloads/myshell/order_judge.c	17
4.10.1	Definições dos tipos	18
4.10.1.1	cmd_fun_t	18
4.10.1.2	CMD_STRUCT	18
4.10.1.3	pCMD_STRUCT	18
4.10.2	Funções	18
4.10.2.1	bgs_fun	18
4.10.2.2	cd_fun	18

4.10.2.3	exit_fun	19
4.10.2.4	fgs_fun	19
4.10.2.5	fun_shell	19
4.10.2.6	help_fun	19
4.10.2.7	history_fun	19
4.10.2.8	job_fun	19
4.10.2.9	pause_fun	19
4.10.3	Variáveis	19
4.10.3.1	cmd_list	19
4.10.3.2	OLD_DIR	20
4.11	Referência do Arquivo Downloads/myshell/order_judge.h	20
4.11.1	Definições e macros	20
4.11.1.1	max_buf	20
4.11.2	Funções	20
4.11.2.1	fun_shell	20
4.12	Referência do Arquivo Downloads/myshell/record_history.c	20
4.12.1	Definições dos tipos	21
4.12.1.1	ENV_HISTORY	21
4.12.2	Funções	21
4.12.2.1	add_history	21
4.12.2.2	get_cmd	22
4.12.2.3	history	22
4.12.2.4	history_list	22
4.12.3	Variáveis	22
4.12.3.1	cur_his_num	22
4.12.3.2	envhis	22
4.13	Referência do Arquivo Downloads/myshell/record_history.h	22
4.13.1	Funções	23
4.13.1.1	add_history	23
4.13.1.2	get_cmd	23
4.13.1.3	history	23
4.13.1.4	history_list	24
4.14	Referência do Arquivo Downloads/myshell/redirect.c	24
4.14.1	Funções	24

4.14.1.1	redirect	24
4.14.1.2	reset_io	25
4.14.2	Variáveis	25
4.14.2.1	fd_bak	25
4.15	Referência do Arquivo Downloads/myshe/redirect.h	25
4.15.1	Funções	25
4.15.1.1	redirect	25
4.15.1.2	reset_io	26

Capítulo 1

Índice dos Componentes

1.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

<code>cmd_info</code>	5
<code>ENV_HISTORY</code>	5
<code>NODE</code>	6

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Downloads/myshell/en_fou.c	7
Downloads/myshell/en_fou.h	8
Downloads/myshell/jobs.c	9
Downloads/myshell/jobs.h	11
Downloads/myshell/keypress.c	13
Downloads/myshell/keypress.h	14
Downloads/myshell/main.c	14
Downloads/myshell/my_pipe.c	16
Downloads/myshell/my_pipe.h	16
Downloads/myshell/order_judge.c	17
Downloads/myshell/order_judge.h	20
Downloads/myshell/record_history.c	20
Downloads/myshell/record_history.h	22
Downloads/myshell/redirect.c	24
Downloads/myshell/redirect.h	25

Capítulo 3

Classes

3.1 Referência da Estrutura `cmd_info`

Atributos Públicos

- `char * cmd_name`
- `cmd_fun_t fun`

3.1.1 Atributos

3.1.1.1 `char* cmd_info::cmd_name`

3.1.1.2 `cmd_fun_t cmd_info::fun`

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- Downloads/myshell/[order_judge.c](#)

3.2 Referência da Estrutura `ENV_HISTORY`

Atributos Públicos

- `int start`
- `int end`
- `char his_cmd [120][100]`

3.2.1 Atributos

3.2.1.1 `int ENV_HISTORY::end`

3.2.1.2 `char ENV_HISTORY::his_cmd[120][100]`

3.2.1.3 `int ENV_HISTORY::start`

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- Downloads/myshell/[record_history.c](#)

3.3 Referência da Estrutura NODE

```
#include <jobs.h>
```

Atributos Públicos

- `int num`
- `pid_t pid`
- `char cmd[100]`
- `char state[10]`
- `struct NODE * next`

3.3.1 Atributos

3.3.1.1 `char NODE::cmd[100]`

3.3.1.2 `struct NODE* NODE::next`

3.3.1.3 `int NODE::num`

3.3.1.4 `pid_t NODE::pid`

3.3.1.5 `char NODE::state[10]`

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- Downloads/myshell/[jobs.h](#)

Capítulo 4

Arquivos

4.1 Referência do Arquivo Downloads/myshell/en_fou.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "en_fou.h"
```

Funções

- void `init_environ` ()
Define o perfil do ambiente.
- char * `is_founded` (char *cmd)
Verifica se o comando é interno (implementado) ou não.

Variáveis

- static char `env_buf` [555] = ""
- char * `env_argv` [20] = {NULL}
- char `cur_cmd` [200] = ""

4.1.1 Funções

4.1.1.1 void `init_environ` ()

Define o perfil do ambiente.

Abre o arquivo `mysh_profile`, que tem o caminho do diretório principal.

Garante que env_buf pegou o caminho do arquivo.

Dividir os outros argumentos.

4.1.1.2 char* is_founded (char * cmd)

Verifica se o comando é interno (implementado) ou não.

Retorna

Retorna o comando

Verifica permissões para todos os argumentos.

4.1.2 Variáveis

4.1.2.1 char cur_cmd[200] = ""

4.1.2.2 char* env_argv[20] = {NULL}

4.1.2.3 char env_buf[555] = "" [static]

4.2 Referência do Arquivo Downloads/myshell/en_fou.h

Funções

- void [init_environ](#) ()
Define o perfil do ambiente.
- char * [is_founded](#) (char *cmd)
Verifica se o comando é interno (implementado) ou não.

4.2.1 Funções

4.2.1.1 void [init_environ](#) ()

Define o perfil do ambiente.

Abre o arquivo mysh_profile, que tem o caminho do diretório principal.

Garante que env_buf pegou o caminho do arquivo.

Dividir os outros argumentos.

4.2.1.2 char* [is_founded](#) (char * cmd)

Verifica se o comando é interno (implementado) ou não.

Retorna

Retorna o comando

Verifica permissões para todos os argumentos.

4.3 Referência do Arquivo Downloads/myshell/jobs.c

```
#include "jobs.h"
```

Funções

- void `ctrl_z` ()
CTRL + Z.
- void `ctrl_c` ()
CTRL + C.
- void `sig_init` ()
Inicializa os sinais.
- `NODE * find_node` (pid_t pid)
Achar o job.
- void `del_node` (void)
Deletar um job na lista.
- void `add_job_node` (int status, pid_t pid, char *cmd_name)
Adicionar um job na lista.
- void `jobs_fun` ()
Lista todos os jobs na lista.
- void `bg_fun` (char *job_num)
Coloca processo em background.
- void `fg_fun` (char *job_num)
Coloca processo em foreground.

Variáveis

- `NODE * head`
- `NODE * end`
- `NODE * p`
- `NODE * q`
- int `fg_pid` = 0

4.3.1 Funções

4.3.1.1 `void add_job_node (int status, pid_t pid, char * cmd_name)`

Adicionar um job na lista.

Verifica se status = 123, flag que indica se o processo rodará no background ou não.

Se cabeça = NULL, adiciona o primeiro job na lista.

Se for um novo processo, adiciona no final da lista.

Senão, setar o status do processo como STOPPED.

Se for encontrado um &, inicializa processo no background.

Senão, para o processo.

4.3.1.2 `void bg_fun (char * job_num)`

Coloca processo em background.

4.3.1.3 `void ctrl_c ()`

CTRL + C.

Remove o job na lista.

4.3.1.4 `void ctrl_z ()`

CTRL + Z.

4.3.1.5 `void del_node (void)`

Deletar um job na lista.

4.3.1.6 `void fg_fun (char * job_num)`

Coloca processo em foreground.

4.3.1.7 `NODE* find_node (pid_t pid)`

Achar o job.

Retorna

Retorna o pid do nó

4.3.1.8 void jobs_fun (void)

Lista todos os jobs na lista.

4.3.1.9 void sig_init (void)

Inicializa os sinais.

4.3.2 Variáveis

4.3.2.1 NODE * end

4.3.2.2 int fg_pid = 0

4.3.2.3 NODE* head

4.3.2.4 NODE * p

4.3.2.5 NODE * q

4.4 Referência do Arquivo Downloads/mysheIl/jobs.h

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <sys/wait.h>
```

Componentes

- struct [NODE](#)

Definições de Tipos

- typedef struct [NODE](#) [NODE](#)

Funções

- void [add_job_node](#) (int status, pid_t pid, char *cmd_name)
Adicionar um job na lista.
- void [del_node](#) (void)
Deletar um job na lista.

- void `sig_init` (void)
Inicializa os sinais.
- void `bg_fun` (char *`job_num`)
Coloca processo em background.
- void `jobs_fun` (void)
Lista todos os jobs na lista.
- void `fg_fun` (char *`job_num`)
Coloca processo em foreground.

Variáveis

- int `fg_pid`
- int `bg_pid`

4.4.1 Definições dos tipos

4.4.1.1 typedef struct `NODE` `NODE`

4.4.2 Funções

4.4.2.1 void `add_job_node` (int `status`, pid_t `pid`, char * `cmd_name`)

Adicionar um job na lista.

Verifica se `status` = 123, flag que indica se o processo rodará no background ou não.

Se `cabeça` = NULL, adiciona o primeiro job na lista.

Se for um novo processo, adiciona no final da lista.

Senão, setar o status do processo como STOPPED.

Se for encontrado um &, inicializa processo no background.

Senão, para o processo.

4.4.2.2 void `bg_fun` (char * `job_num`)

Coloca processo em background.

4.4.2.3 void `del_node` (void)

Deletar um job na lista.

4.4.2.4 void `fg_fun` (char * `job_num`)

Coloca processo em foreground.

4.4.2.5 void jobs_fun (void)

Lista todos os jobs na lista.

4.4.2.6 void sig_init (void)

Inicializa os sinais.

4.4.3 Variáveis

4.4.3.1 int bg_pid

4.4.3.2 int fg_pid

4.5 Referência do Arquivo Downloads/myshell/keypress.c

```
#include "keypress.h"
```

Funções

- void [set_keypress](#) (void)
Define o modo para pegar caracter por caracter do teclado.
- void [reset_keypress](#) (void)
Resetar o teclado.

Variáveis

- static struct termios [stored_settings](#)

4.5.1 Funções

4.5.1.1 void reset_keypress (void)

Resetar o teclado.

Mudança ocorre imediatamente.

4.5.1.2 void set_keypress (void)

Define o modo para pegar caracter por caracter do teclado.

Setar os parâmetros associados com o terminal.

Desativa modo canônico, e define o tamanho do buffer para 1 byte.

4.5.2 Variáveis

4.5.2.1 `struct termios stored_settings` `[static]`

4.6 Referência do Arquivo Downloads/myshell/keypress.h

```
#include <termios.h>
```

Funções

- void `set_keypress` (void)
Define o modo para pegar caracter por caracter do teclado.
- void `reset_keypress` (void)
Resetar o teclado.

4.6.1 Funções

4.6.1.1 void `reset_keypress` (void)

Resetar o teclado.

Mudança ocorre imediatamente.

4.6.1.2 void `set_keypress` (void)

Define o modo para pegar caracter por caracter do teclado.

Setar os parâmetros associados com o terminal.

Desativa modo canônico, e define o tamanho do buffer para 1 byte.

4.7 Referência do Arquivo Downloads/myshell/main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include "order_judge.h"
```

```
#include "en_fou.h"
#include "redirect.h"
#include "my_pipe.h"
#include "record_history.h"
#include "keypress.h"
#include "jobs.h"
```

Funções

- int `main` (int argc, char *argv[])

4.7.1 Funções

4.7.1.1 int main (int argc, char * argv[])

Inicializa os sinais, tais como Ctrl_z e Ctrl_c.

Define o ambiente.

Imprime o caminho atual.

Lê do teclado e armazena em cmd_buf.

Se for rodar no background, remove '&' de cmd_buf e seta flag de bg para 1.

Se for um programa executável, seta a flag de executável para 1.

Verifica se há pipe.

Coloca o conteúdo de cmd_buf como argumentos da chamada.

Se for um comando interno chama reset_io.

Se for um comando implementado, dá fork no processo.

Se houver erro ao criar processo filho, printa uma mensagem.

Trata erros dos programas executáveis, acesso não permitido, etc.

Se não for background, espera o término do processo filho.

Se for background, seta o flag de bg para 0 e o status para 123.

Adiciona o job na lista dos jobs.

Senão, exibe mensagem de "Comando inválido".

Limpas os buffers de E/S

Espera o processo terminar

4.8 Referência do Arquivo Downloads/myshell/my_pipe.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include "my_pipe.h"
#include <wait.h>
#include "redirect.h"
#include "en_fou.h"
```

Funções

- int `my_pipe` (char *cmd_buf)
Verifica se há pipe ou não.

4.8.1 Funções

4.8.1.1 int my_pipe (char * cmd_buf)

Verifica se há pipe ou não.

Retorna

1 se há pipe, 0 se não há pipe

Se fork = 0 (processo filho).

Substitui o output padrão por fds[1].

Executa se o comando estiver implementado.

Senão, se fork = 0 (processo filho).

Substitui o input padrão por fds[0].

Executa se o comando estiver implementado.

Senão, processo pai.

4.9 Referência do Arquivo Downloads/myshell/my_pipe.h

Funções

- int `my_pipe` (char *cmd_buf)

Verifica se há pipe ou não.

4.9.1 Funções

4.9.1.1 `int my_pipe (char * cmd_buf)`

Verifica se há pipe ou não.

Retorna

1 se há pipe, 0 se não há pipe

Se `fork = 0` (processo filho).

Substitui o output padrão por `fds[1]`.

Executa se o comando estiver implementado.

Senão, se `fork = 0` (processo filho).

Substitui o input padrão por `fds[0]`.

Executa se o comando estiver implementado.

Senão, processo pai.

4.10 Referência do Arquivo Downloads/myshell/order_judge.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "order_judge.h"
#include "record_history.h"
#include "jobs.h"
```

Componentes

- struct `cmd_info`

Definições de Tipos

- `typedef int(* cmd_fun_t)(int argc, char *argv[])`
- `typedef struct cmd_info CMD_STRUCT`
- `typedef struct cmd_info pCMD_STRUCT`

Funções

- int `cd_fun` (int argc, char *argv[])
Comando cd.
- int `exit_fun` (int argc, char *argv[])
Comando exit.
- int `pause_fun` (int argc, char *argv[])
Comando pause.
- int `help_fun` (int argc, char *argv[])
Comando Help.
- int `job_fun` (int argc, char *argv[])
Comando jobs.
- int `history_fun` (int argc, char *argv[])
Comando history.
- int `fgs_fun` (int argc, char *argv[])
Comando fg.
- int `bgs_fun` (int argc, char *argv[])
Comando bg.
- int `fun_shell` (int argc, char *argv[])
Executa o comando digitado (se válido)

Variáveis

- char `OLD_DIR` [100] = ""
- `CMD_STRUCT` `cmd_list` []
Lista dos comandos implementados.

4.10.1 Definições dos tipos

4.10.1.1 `typedef int(* cmd_fun_t)(int argc, char *argv[])`

4.10.1.2 `typedef struct cmd_info CMD_STRUCT`

4.10.1.3 `typedef struct cmd_info pCMD_STRUCT`

4.10.2 Funções

4.10.2.1 `int bgs_fun (int argc, char * argv[])`

Comando bg.

4.10.2.2 `int cd_fun (int argc, char * argv[])`

Comando cd.

4.10.2.3 `int exit_fun (int argc, char * argv[])`

Comando exit.

4.10.2.4 `int fgs_fun (int argc, char * argv[])`

Comando fg.

4.10.2.5 `int fun_shell (int argc, char * argv[])`

Executa o comando digitado (se válido)

Retorna

Se válido, retorna 1, 0 caso contrário

4.10.2.6 `int help_fun (int argc, char * argv[])`

Comando Help.

Abre o arquivo de ajuda.

Procura por palavras-chave.

Se o arquivo ainda estiver aberto quando a função for retornar, fechá-lo.

4.10.2.7 `int history_fun (int argc, char * argv[])`

Comando history.

4.10.2.8 `int job_fun (int argc, char * argv[])`

Comando jobs.

4.10.2.9 `int pause_fun (int argc, char * argv[])`

Comando pause.

4.10.3 Variáveis

4.10.3.1 `CMD_STRUCT cmd_list[]`

Valor Inicial:

```
{
    {"cd", cd_fun},
    {"exit", exit_fun},
        {"help", help_fun},
        {"history", history_fun},
        {"pause", pause_fun},
        {"jobs", job_fun},
        {"fg", fgs_fun},
    {"bg", bgs_fun}
}
```

Lista dos comandos implementados.

CMD_STRUCT cmd_list

4.10.3.2 char OLD_DIR[100] = ""

4.11 Referência do Arquivo Downloads/myshell/order_judge.h

Definições e Macros

- #define max_buf 1024

Funções

- int fun_shell (int argc, char *argv[])
Executa o comando digitado (se válido)

4.11.1 Definições e macros

4.11.1.1 #define max_buf 1024

4.11.2 Funções

4.11.2.1 int fun_shell (int argc, char * argv[])

Executa o comando digitado (se válido)

Retorna

Se válido, retorna 1, 0 caso contrário

4.12 Referência do Arquivo Downloads/myshell/record_history.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include "keypress.h"
```

Componentes

- struct [ENV_HISTORY](#)

Definições de Tipos

- typedef struct [ENV_HISTORY](#) [ENV_HISTORY](#)

Funções

- void [add_history](#) (char *cmd)
Adiciona comando digitado ao histórico.
- int [history](#) (int key, char *cmd)
- void [history_list](#) (int i)
Listar o histórico.
- void [get_cmd](#) (char *cmd, int size)
Lê as teclas do teclado e analisa-as.

Variáveis

- [ENV_HISTORY](#) [envhis](#) = {0,0,{""}}}
- int [cur_his_num](#) = 0

4.12.1 Definições dos tipos

4.12.1.1 typedef struct [ENV_HISTORY](#) [ENV_HISTORY](#)

4.12.2 Funções

4.12.2.1 void [add_history](#) (char * *cmd*)

Adiciona comando digitado ao histórico.

Inicializa os históricos com nulo.

Adiciona comando no vetor de histórico.

Como tem tamanho 120, divide-o e adiciona ao final dele.

Se final igual ao começo, incrementa o começo.

Seta a posição atual no final do histórico.

4.12.2.2 void get_cmd (char * cmd, int size)

Lê as teclas do teclado e analisa-as.

Lê do teclado.

Se RIGHT_ARROW.

Senão, se CRTZ + Z.

Senão, se CRTL + C.

Senão, se ENTER.

Senão, se BACKSPACE.

Senão for nenhuma tecla acima mencionada, continue.

Seta as mudanças imediatamente.

Chega ao fim da string.

Coloca no histórico.

4.12.2.3 int history (int key, char * cmd)

Se UP_ARROW, mostra os conteúdos anteriores de baixo para cima a partir da posição atual.

Possível enquanto não for o início.

Se DOWN_ARROW, mostra os conteúdos anteriores de cima para baixo a partir da posição atual

Possível enquanto não for o fim.

Limpa o que tem no terminal e reimprime o caminho.

Imprime o comando no histórico.

4.12.2.4 void history_list (int i)

Listar o histórico.

4.12.3 Variáveis

4.12.3.1 int cur_his_num = 0

4.12.3.2 ENV_HISTORY envhis = { 0, 0, { "" } }

4.13 Referência do Arquivo Downloads/myshell/record_history.h

Funções

- void `add_history` (char *cmd)
Adiciona comando digitado ao histórico.
- int `history` (int key, char *cmd)
- void `history_list` (int i)
Listar o histórico.
- void `get_cmd` (char *cmd, int size)
Lê as teclas do teclado e analisa-as.

4.13.1 Funções

4.13.1.1 void add_history (char * cmd)

Adiciona comando digitado ao histórico.

Inicializa os históricos com nulo.

Adiciona comando no vetor de histórico.

Como tem tamanho 120, divide-o e adiciona ao final dele.

Se final igual ao começo, incrementa o começo.

Seta a posição atual no final do histórico.

4.13.1.2 void get_cmd (char * cmd, int size)

Lê as teclas do teclado e analisa-as.

Lê do teclado.

Se RIGHT_ARROW.

Senão, se CRTZ + Z.

Senão, se CRTL + C.

Senão, se ENTER.

Senão, se BACKSPACE.

Senão for nenhuma tecla acima mencionada, continue.

Seta as mudanças imediatamente.

Chega ao fim da string.

Coloca no histórico.

4.13.1.3 int history (int key, char * cmd)

Se UP_ARROW, mostra os conteúdos anteriores de baixo para cima a partir da posição atual.

Possível enquanto não for o início.

Se `DOWN_ARROW`, mostra os conteúdos anteriores de cima para baixo a partir da posição atual

Possível enquanto não for o fim.

Limpa o que tem no terminal e reimprime o caminho.

Imprime o comando no histórico.

4.13.1.4 void history_list (int i)

Listar o histórico.

4.14 Referência do Arquivo Downloads/myshell/redirect.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include "redirect.h"
```

Funções

- int `redirect` (char *str, char *argv[])
- int `reset_io` ()

Reseta E/S.

Variáveis

- int `fd_bak` [2] = {100,101}

4.14.1 Funções

4.14.1.1 int redirect (char * str, char * argv[])

Trata '>'.
Se '>': coloca cursor no fim do arquivo

Trata '<'.
Se `wfile != NULL`, abre arquivo de escrita com as flags definidas acima.

Substitui fd_bak[1] pelo output padrão.

Substitui o output padrão por wfd.

Se rfile != NULL, abre o arquivo para leitura.

Substitui fd_bak[0] pelo input padrão.

Substitui o input padrão por rf.

4.14.1.2 int reset_io ()

Reseta E/S.

Substitui o output padrão por fd_bak[1].

Substitui o input padrão por fd_bak[0].

4.14.2 Variáveis

4.14.2.1 int fd_bak[2] = { 100,101 }

4.15 Referência do Arquivo Downloads/myshell/redirect.h

Funções

- int [redirect](#) (char *str, char *argv[])
- int [reset_io](#) ()

Reseta E/S.

4.15.1 Funções

4.15.1.1 int redirect (char * str, char * argv[])

Trata '>'.

Se '>': coloca cursor no fim do arquivo

Trata '<'.

Se wfile != NULL, abre arquivo de escrita com as flags definidas acima.

Substitui fd_bak[1] pelo output padrão.

Substitui o output padrão por wfd.

Se rfile != NULL, abre o arquivo para leitura.

Substitui fd_bak[0] pelo input padrão.

Substitui o input padrão por rf.

4.15.1.2 int reset_io ()

Reseta E/S.

Substitui o output padrão por fd_bak[1].

Substitui o input padrão por fd_bak[0].

Índice Remissivo

- add_history
 - record_history.c, [21](#)
 - record_history.h, [23](#)
- add_job_node
 - jobs.c, [10](#)
 - jobs.h, [12](#)
- bg_fun
 - jobs.c, [10](#)
 - jobs.h, [12](#)
- bg_pid
 - jobs.h, [13](#)
- bgs_fun
 - order_judge.c, [18](#)
- cd_fun
 - order_judge.c, [18](#)
- cmd
 - NODE, [6](#)
- cmd_fun_t
 - order_judge.c, [18](#)
- cmd_info, [5](#)
 - cmd_name, [5](#)
 - fun, [5](#)
- cmd_list
 - order_judge.c, [19](#)
- cmd_name
 - cmd_info, [5](#)
- CMD_STRUCT
 - order_judge.c, [18](#)
- ctrl_c
 - jobs.c, [10](#)
- ctrl_z
 - jobs.c, [10](#)
- cur_cmd
 - en_fou.c, [8](#)
- cur_his_num
 - record_history.c, [22](#)
- del_node
 - jobs.c, [10](#)
- jobs.h, [12](#)
- Downloads/myshell/en_fou.c, [7](#)
- Downloads/myshell/en_fou.h, [8](#)
- Downloads/myshell/jobs.c, [9](#)
- Downloads/myshell/jobs.h, [11](#)
- Downloads/myshell/keypress.c, [13](#)
- Downloads/myshell/keypress.h, [14](#)
- Downloads/myshell/main.c, [14](#)
- Downloads/myshell/my_pipe.c, [16](#)
- Downloads/myshell/my_pipe.h, [16](#)
- Downloads/myshell/order_judge.c, [17](#)
- Downloads/myshell/order_judge.h, [20](#)
- Downloads/myshell/record_history.c, [20](#)
- Downloads/myshell/record_history.h, [22](#)
- Downloads/myshell/redirect.c, [24](#)
- Downloads/myshell/redirect.h, [25](#)
- en_fou.c
 - cur_cmd, [8](#)
 - env_argv, [8](#)
 - env_buf, [8](#)
 - init_envron, [7](#)
 - is_founded, [8](#)
- en_fou.h
 - init_envron, [8](#)
 - is_founded, [8](#)
- end
 - ENV_HISTORY, [5](#)
 - jobs.c, [11](#)
- env_argv
 - en_fou.c, [8](#)
- env_buf
 - en_fou.c, [8](#)
- ENV_HISTORY, [5](#)
 - end, [5](#)
 - his_cmd, [5](#)
 - record_history.c, [21](#)
 - start, [6](#)
- envhis
 - record_history.c, [22](#)
- exit_fun

- order_judge.c, 18
- fd_bak
 - redirect.c, 25
- fg_fun
 - jobs.c, 10
 - jobs.h, 12
- fg_pid
 - jobs.c, 11
 - jobs.h, 13
- fgs_fun
 - order_judge.c, 19
- find_node
 - jobs.c, 10
- fun
 - cmd_info, 5
- fun_shell
 - order_judge.c, 19
 - order_judge.h, 20
- get_cmd
 - record_history.c, 22
 - record_history.h, 23
- head
 - jobs.c, 11
- help_fun
 - order_judge.c, 19
- his_cmd
 - ENV_HISTORY, 5
- history
 - record_history.c, 22
 - record_history.h, 23
- history_fun
 - order_judge.c, 19
- history_list
 - record_history.c, 22
 - record_history.h, 24
- init_envron
 - en_fou.c, 7
 - en_fou.h, 8
- is_founded
 - en_fou.c, 8
 - en_fou.h, 8
- job_fun
 - order_judge.c, 19
- jobs.c
 - add_job_node, 10
 - bg_fun, 10
- ctrl_c, 10
- ctrl_z, 10
- del_node, 10
- end, 11
- fg_fun, 10
- fg_pid, 11
- find_node, 10
- head, 11
- jobs_fun, 10
- p, 11
- q, 11
- sig_init, 11
- jobs.h
 - add_job_node, 12
 - bg_fun, 12
 - bg_pid, 13
 - del_node, 12
 - fg_fun, 12
 - fg_pid, 13
 - jobs_fun, 12
 - NODE, 12
 - sig_init, 13
- jobs_fun
 - jobs.c, 10
 - jobs.h, 12
- keypress.c
 - reset_keypress, 13
 - set_keypress, 13
 - stored_settings, 14
- keypress.h
 - reset_keypress, 14
 - set_keypress, 14
- main
 - main.c, 15
- main.c
 - main, 15
- max_buf
 - order_judge.h, 20
- my_pipe
 - my_pipe.c, 16
 - my_pipe.h, 17
- my_pipe.c
 - my_pipe, 16
- my_pipe.h
 - my_pipe, 17
- next
 - NODE, 6

NODE, 6
 cmd, 6
 jobs.h, 12
 next, 6
 num, 6
 pid, 6
 state, 6
num
 NODE, 6
OLD_DIR
 order_judge.c, 20
order_judge.c
 bgs_fun, 18
 cd_fun, 18
 cmd_fun_t, 18
 cmd_list, 19
 CMD_STRUCT, 18
 exit_fun, 18
 fgs_fun, 19
 fun_shell, 19
 help_fun, 19
 history_fun, 19
 job_fun, 19
 OLD_DIR, 20
 pause_fun, 19
 pCMD_STRUCT, 18
order_judge.h
 fun_shell, 20
 max_buf, 20
p
 jobs.c, 11
pause_fun
 order_judge.c, 19
pCMD_STRUCT
 order_judge.c, 18
pid
 NODE, 6
q
 jobs.c, 11
record_history.c
 add_history, 21
 cur_his_num, 22
 ENV_HISTORY, 21
 envhis, 22
 get_cmd, 22
 history, 22
 history_list, 22
record_history.h
 add_history, 23
 get_cmd, 23
 history, 23
 history_list, 24
redirect
 redirect.c, 24
 redirect.h, 25
redirect.c
 fd_bak, 25
 redirect, 24
 reset_io, 25
redirect.h
 redirect, 25
 reset_io, 25
reset_io
 redirect.c, 25
 redirect.h, 25
reset_keypress
 keypress.c, 13
 keypress.h, 14
set_keypress
 keypress.c, 13
 keypress.h, 14
sig_init
 jobs.c, 11
 jobs.h, 13
start
 ENV_HISTORY, 6
state
 NODE, 6
stored_settings
 keypress.c, 14