fescape

1.0

Generated by Doxygen 1.10.0

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 actions.h

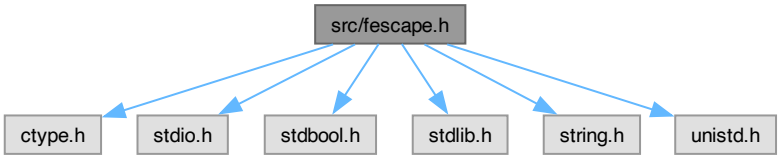
```
00001 #ifndef ACTIONS_H
00002 #define ACTIONS_H
00003
00008 typedef enum {
00009     ACTION_PREP,
00010     ACTION_RESTORE,
00011     ACTION_ADD,
00012     ACTION_FLUSH,
00013     ACTION_INVALID
00014 } Action;
00015
00016 #endif /* ACTIONS_H */
```

2.2 src/fescape.h File Reference

Filter unprintable characters from input stream.

```
#include <ctype.h>
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

Include dependency graph for fescape.h:



Functions

- void usage (const char *program)

  Display help to user.
- void fescape (FILE *input_stream, FILE *output_stream, bool repeats, bool octal)

  convert non-ASCII characters to hex or octal representation

2.2.1 Detailed Description

Filter unprintable characters from input stream.

Author

Robert Primmer (https://github.com/rprimmer)

Files that contain non-printable characters mess up the display when printed (e.g., via cat(1)). This program allows the display of such files, substituting hex (or optionally octal) codes for the non-printable characters. Optionally it can show the count for repeated non-printable characters rather than display each repeated hex/octal code.

Version

1.0

Date

2024-03-24

2.2.2 Function Documentation

2.2.2.1 fescape()

```
void fescape (
            FILE * input_stream,
            FILE * output_stream,
            bool repeats,
            bool octal )
```

convert non-ASCII characters to hex or octal representation

Parameters

input_stream	Input stream to read.
output_stream	Output stream to write.
repeats	If true, display repeated character count.
octal	If true, display control sequences in octal instead of hex.

2.2.2.2 usage()

```
void usage (
            const char * program )
```

Display help to user.

Parameters

program	Calling program name

2.3 fescape.h

Go to the documentation of this file.

```
00001
00014 #pragma once
00015
00016 #include <ctype.h>
00017 #include <stdio.h>
00018 #include <stdbool.h>
00019 #include <stdlib.h>
00020 #include <string.h>
00021 #include <unistd.h>
00022
00028 void usage(const char *program);
00029
00038 void fescape(FILE *input_stream, FILE *output_stream, bool repeats, bool octal);
```

2.4 src/system-actions.h File Reference

Common functions and system actions.

```
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <fnmatch.h>
#include <grp.h>
#include <libgen.h>
#include <limits.h>
#include <pwd.h>
#include <regex.h>
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for system-actions.h:

Macros

- #define **HANDLE_ERROR**(fmt, ...) handleError(true, __FILE__, __func__, __LINE__, fmt, ##__VA_ARGS↩ __)
- #define **REPORT_ERROR**(fmt, ...) handleError(false, __FILE__, __func__, __LINE__, fmt, ##__VA_ARGS↩ __)

Functions

- void handleError (bool fatal, char *file, const char *func, int line, const char *fmt,...)

  Common error handling routine.
- int booleanQuery (const char *prompt)

  Query user for yes or no.
- int fileExists (const char *filename)

  Check for file existence.
- int copyFile (const char *src, const char *dest)

  Make a copy of a file. Uses fread(3) & fwrite(3).
- int copyFile2 (const char *src, const char *dest)

  Make a copy of a file. Uses read(2) & write(2).
- int lsFiles (const char *dirname, const char *files)

  List files in a directory.
- int fileInfo (const char *filepath)

  Display information about a file.
- int checkProcess (const char *process_name)

  Check if a process is currently running.
- int displayProcess (const char *process_name)

  Display info on a running process.
- int validateDNSname (const char *dns_name)

  DNS name must start & end with a letter or a number and can only contain letters, numbers, and hyphens.

2.4.1 Detailed Description

Common functions and system actions.

Author

Robert Primmer (https://github.com/rprimmer)

Version

1.2

Date

2024-03-24

2.4.2 Function Documentation

2.4.2.1 booleanQuery()

```
int booleanQuery (
            const char * prompt )
```

Query user for yes or no.

Parameters

prompt	Message to be displayed to user.

Returns

int Return true if user entered y or Y.

2.4.2.2 checkProcess()

```
int checkProcess (
            const char * process_name )
```

Check if a process is currently running.

Parameters

process_name	Process to look for.

Returns

int Return status.

2.4.2.3 copyFile()

```
int copyFile (
            const char * src,
            const char * dest )
```

Make a copy of a file. Uses fread(3) & fwrite(3).

Parameters

src	File to be copied.
dest	Filename of copy.

Returns

int Return status.

2.4.2.4 copyFile2()

```
int copyFile2 (
            const char * src,
            const char * dest )
```

Make a copy of a file. Uses read(2) & write(2).

Parameters

src	File to be copied.
dest	Filename of copy.

Returns

int Return status.

2.4.2.5 displayProcess()

```
int displayProcess (
            const char * process_name )
```

Display info on a running process.

Parameters

process_name	Process to look for.

Returns

int Return status.

2.4.2.6 fileExists()

```
int fileExists (
            const char * filename )
```

Check for file existence.

Parameters

filename	File to check.

Returns

int Return true of file exists.

2.4.2.7 fileInfo()

```
int fileInfo (
            const char * filepath )
```

Display information about a file.

Parameters

filepath	File to stat.

Returns

int Return status.

2.4.2.8 handleError()

```
void handleError (
             bool fatal,
             char * file,
             const char * func,
             int line,
             const char * fmt,
              ...  )
```

Common error handling routine.

Parameters

fatal	If true, exit program, else returns to the caller.
file	C filename (translation unit) of caller.
func	Function name of caller.
line	Line number in translation unit.
fmt	Optional parameters can be provided (va_list).

2.4.2.9 lsFiles()

```
int lsFiles (
             const char * dirname,
             const char * files )
```

List files in a directory.

Parameters

dirname	Directory housing files.
files	Files to list.

Returns

int Return status.

2.4.2.10 validateDNSname()

```
int validateDNSname (
             const char * dns_name )
```

DNS name must start & end with a letter or a number and can only contain letters, numbers, and hyphens.

Parameters

dns_name	DNS name to check.

Returns

int Return status.

2.5 system-actions.h

Go to the documentation of this file.

```
00001
00009 #ifndef SYSTEM_ACTIONS_H
00010 #define SYSTEM_ACTIONS_H
00011
00012 #include <dirent.h>
00013 #include <errno.h>
00014 #include <fcntl.h>
00015 #include <fnmatch.h>
00016 #include <grp.h>
00017 #include <libgen.h>
00018 #include <limits.h>
00019 #include <pwd.h>
00020 #include <regex.h>
00021 #include <stdarg.h>
00022 #include <stdbool.h>
00023 #include <stdio.h>
00024 #include <stdlib.h>
00025 #include <string.h>
00026 #include <sys/stat.h>
00027 #include <sys/types.h>
00028 #include <sys/wait.h>
00029 #include <time.h>
00030 #include <unistd.h>
00031
00032 // ##__VA_ARGS__ is a GNU extension that still works if __VA_ARGS__ is empty,
00033 // which supports calling the macro with just a string or with additional format arguments.
00034 // Modern compilers support this so I didn't want to clutter the code with a bunch of
00035 // #ifdef __GNUC__ conditionals just for the sake of some ancient compiler from long long ago.
00036 // __func__ was introduced in C99.
00037 #define HANDLE_ERROR(fmt, ...) handleError(true, __FILE__, __func__, __LINE__, fmt, ##__VA_ARGS__)
00038 #define REPORT_ERROR(fmt, ...) handleError(false, __FILE__, __func__, __LINE__, fmt, ##__VA_ARGS__)
00039
00049 void handleError(bool fatal, char *file, const char *func, int line, const char *fmt, ...);
00050
00057 int booleanQuery(const char *prompt);
00058
00065 int fileExists(const char *filename);
00066
00074 int copyFile(const char *src, const char *dest);
00075
00083 int copyFile2(const char *src, const char *dest);
00084
00092 int lsFiles(const char *dirname, const char *files);
00093
00100 int fileInfo(const char *filepath);
00101
00108 int checkProcess(const char *process_name);
00109
00116 int displayProcess(const char *process_name);
00117
00124 int validateDNSname(const char *dns_name);
00125
00126 #endif /* SYSTEM_ACTIONS_H */
```

Index