

Including uAPI header files

Sometimes, it is useful to include header files and C example codes in order to describe the userspace API and to generate cross-references between the code and the documentation. Adding cross-references for userspace API files has an additional vantage: Sphinx will generate warnings if a symbol is not found at the documentation. That helps to keep the uAPI documentation in sync with the Kernel changes. The `.ref: parse_headers.pl <parse_headers>` provide a way to generate such cross-references. It has to be called via Makefile, while building the documentation. Please see [Documentation/userspace-api/media/Makefile](#) for an example about how to use it inside the Kernel tree.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\doc-guide\[linux-master] [Documentation] [doc-guide]parse-headers.rst, line 5); *backlink*
Unknown interpreted text role "ref".

parse_headers.pl

NAME

parse_headers.pl - parse a C file, in order to identify functions, structs, enums and defines and create cross-references to a Sphinx book.

SYNOPSIS

parse_headers.pl [`<options>`] `<C_FILE>` `<OUT_FILE>` [`<EXCEPTIONS_FILE>`]

Where `<options>` can be: `--debug`, `--help` or `--usage`.

OPTIONS

--debug

Put the script in verbose mode, useful for debugging.

--usage

Prints a brief help message and exits.

--help

Prints a more detailed help message and exits.

DESCRIPTION

Convert a C header or source file (`C_FILE`), into a ReStructured Text included via `..parsed-literal` block with cross-references for the documentation files that describe the API. It accepts an optional `EXCEPTIONS_FILE` with describes what elements will be either ignored or be pointed to a non-default reference.

The output is written at the (`OUT_FILE`).

It is capable of identifying defines, functions, structs, typedefs, enums and enum symbols and create cross-references for all of them. It is also capable of distinguish `#define` used for specifying a Linux ioctl.

The `EXCEPTIONS_FILE` contain two types of statements: **ignore** or **replace**.

The syntax for the ignore tag is:

ignore **type** **name**

The **ignore** means that it won't generate cross references for a **name** symbol of type **type**.

The syntax for the replace tag is:

replace **type** **name** **new_value**

The **replace** means that it will generate cross references for a **name** symbol of type **type**, but, instead of using the default replacement rule, it will use **new_value**.

For both statements, **type** can be either one of the following:

ioctl

The ignore or replace statement will apply to ioctl definitions like:

```
#define VIDIOC_DBG_S_REGISTER_IOW('V', 79, struct v4l2_dbg_register)
```

define

The ignore or replace statement will apply to any other #define found at C_FILE.

typedef

The ignore or replace statement will apply to typedef statements at C_FILE.

struct

The ignore or replace statement will apply to the name of struct statements at C_FILE.

enum

The ignore or replace statement will apply to the name of enum statements at C_FILE.

symbol

The ignore or replace statement will apply to the name of enum value at C_FILE.

For replace statements, **new_value** will automatically use :c.type: references for **typedef**, **enum** and **struct** types. It will use ref: for **ioctl**, **define** and **symbol** types. The type of reference can also be explicitly defined at the replace statement.

EXAMPLES

ignore define _VIDEODEV2_H

Ignore a #define _VIDEODEV2_H at the C_FILE.

ignore symbol PRIVATE

On a struct like:

```
enum foo { BAR1, BAR2, PRIVATE };
```

It won't generate cross-references for **PRIVATE**.

replace symbol BAR1 :c.type: `foo` replace symbol BAR2 :c.type: `foo`

On a struct like:

```
enum foo { BAR1, BAR2, PRIVATE };
```

It will make the BAR1 and BAR2 enum symbols to cross reference the foo symbol at the C domain.

BUGS

Report bugs to Mauro Carvalho Chehab <mchehab@kernel.org>

COPYRIGHT

Copyright (c) 2016 by Mauro Carvalho Chehab <mchehab+samsung@kernel.org>.

License GPLv2: GNU GPL version 2 <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.