Paul Prince's MPX

R1

Generated by Doxygen 1.7.3

Fri Mar 18 2011 01:47:26

Contents

1 Introduction

1.1 Repository

Version-control information is managed by Git, and hosted by GitHub: `https://github.com/pprince/cs450`

1.2 Documentation

Documentation for developers is generated by Doxygen; for detailed information about the files, functions, data structures, etc. that make up MPX and how they relate to each other, refer to:

- "MPX Programmer's Manual"

which can be found in the doc/ directory. Also, in the same directory, you can find the current version of:

- "MPX User's Manual"

Todo

Generally, documentation is incomplete.

Todo

Generally, we need to make lines break cleanly at 80-columns; Doxygen forces such line-breaks on us in the LaTeX output, but our source code frequently uses longer lines (making the PDF version of the developer manual very ugly!

2 Todo List

page Introduction Generally, documentation is incomplete.

Generally, we need to make lines break cleanly at 80-columns; Doxygen forces such line-breaks on us in the LaTeX output, but our source code frequently uses longer lines (making the PDF version of the developer manual very ugly!

File mpx_cmds.c We should typedef structs (particularly struct mpx_command).

3 Module Documentation

3.1 Pager

Brief description of pager group.

Files

- file pager.c

  Provides a pager feature to MPX, like the Unix `more` command.

- file pager.h

  Provides a pager feature to MPX, like the Unix `more` command.

Defines

- #define SCREEN_ROWS 23
  Defines the number of text rows on the MPX screen.
- #define SCREEN_COLS 79
  Defines the number of text columns on the MPX screen.

Functions

- void pager_init (void)
  This function is called before the first line of paged output is printed.
- void pager_stop (void)
  This function is called before the last line of paged output is printed.
- int pager_printf (const char ∗format,...)
  This function replaces `printf()` when paged output is desired.

Variables

- static rows_printed = 0
  Keeps track of how many rows have been printed on the current screen.

3.1.1 Detailed Description

Brief description of pager group. Detailed description of pager group.

3.1.2 Define Documentation

3.1.2.1 #define SCREEN_ROWS 23

Defines the number of text rows on the MPX screen.

3.1.2.2 #define SCREEN_COLS 79

Defines the number of text columns on the MPX screen.

3.1.3 Function Documentation

3.1.3.1 int pager_printf (const char ∗ *format*, ...)

This function replaces `printf()` when paged output is desired.

Use only this function for output to the screen between calls to `pager_init()` and `pager_stop()`. Writing to the terminal with any other routine/method while paging will cause the output to be garbled, or lines to be missed.

This function makes use of two ANSI-standard C features: variable-length argument lists (va_list), and vprintf.

Returns

Returns the number of bytes written, or EOF to indicate that and error occurred.

```
{
        va_list args;
        va_start(args, format);

        /* Pass the format string and the rest of the args onto vprintf. */
        vprintf(format, args);

        va_end(args);
}
```

3.1.4 Variable Documentation

3.1.4.1 rows_printed = 0 [static]

Keeps track of how many rows have been printed on the current screen.

Note that this is a file-static variable, and thus is only accessible inside the `pager.c` file.

4 Data Structure Documentation

4.1 date_rec Struct Reference

Data Fields

- int **month**
- int **day**
- int **year**

The documentation for this struct was generated from the following file:

- mpx/mpx_supt.h

4.2 mpx_command Struct Reference

Node type for a singly-linked list of MPX commands.

```
#include <mpx_cmds.h>
```

Data Fields

- char ∗ **name**
- void(∗ **function**)(int argc, char ∗argv[ ])
- struct mpx_command ∗ **next**

4.2.1 Detailed Description

Node type for a singly-linked list of MPX commands.

The documentation for this struct was generated from the following file:

- mpx/mpx_cmds.h

4.3 params Struct Reference

Data Fields

- int **op_code**
- int **device_id**
- char ∗ **buf_p**
- int ∗ **count_p**

The documentation for this struct was generated from the following file:

- mpx/mpx_supt.c

4.4 pcb_queue_node_t Struct Reference

Data Fields

- struct pcb_queue_node ∗ next

  Pointer to the next PCB node in the queue.

- struct pcb_queue_node ∗ prev

  Pointer to the previous PCB node in the queue.

- pcb_t ∗ pcb

  Pointer to the actual PCB associated with this node.

4.4.1 Field Documentation

4.4.1.1 struct pcb_queue_node∗ next

Pointer to the next PCB node in the queue.

4.4.1.2 struct pcb_queue_node∗ prev

Pointer to the previous PCB node in the queue.

4.4.1.3 pcb_t∗ pcb

Pointer to the actual PCB associated with this node.

The documentation for this struct was generated from the following file:

- mpx/pcb.h

4.5 pcb_queue_t Struct Reference

PCB queue; represents a queue of processes.

```
#include <pcb.h>
```

Data Fields

- pcb_queue_node_t ∗ head

  Pointer to the first element in the queue.
- pcb_queue_node_t ∗ tail

  Pointer to the last element in the queue.
- unsigned int length

  Number of elements in the queue.
- pcb_queue_sort_order_t sort_order

  Specifies how elements in this queue are sorted at insert-time.

4.5.1 Detailed Description

PCB queue; represents a queue of processes.

4.5.2 Field Documentation

4.5.2.1 pcb_queue_node_t∗ head

Pointer to the first element in the queue.

4.5.2.2 pcb_queue_node_t∗ tail

Pointer to the last element in the queue.

4.5.2.3 unsigned int length

Number of elements in the queue.

4.5.2.4 pcb_queue_sort_order_t sort_order

Specifies how elements in this queue are sorted at insert-time.

The documentation for this struct was generated from the following file:

- mpx/pcb.h

4.6 pcb_t Struct Reference

Process control block structure.

```
#include <pcb.h>
```

Data Fields

- char name [MAX_ARG_LEN+1]

  Name of the process (i.e., its argv[0] in unix-speak).

- process_class_t class

  Process class (differentiates applications from system processes.

- int priority

  Process priority.

- process_state_t state

  Process state (Ready, Running, or Blocked).

- unsigned char ∗ stack_top

  Pointer to the top of this processes's stack.

- unsigned char ∗ stack_base

  Pointer to the bottom of this processes's stack.

- int memory_size

  Memory size ...

- unsigned char ∗ load_address

  Load address ...

- unsigned char ∗ exec_address

  Execution address ...

4.6.1 Detailed Description

Process control block structure.

4.6.2 Field Documentation

4.6.2.1 char name[MAX_ARG_LEN+1]

Name of the process (i.e., its argv[0] in unix-speak).

4.6.2.2 process_class_t class

Process class (differentiates applications from system processes.

4.6.2.3 int priority

Process priority.

Higher numerical value = higher priority.

Valid values are -128 through 127 (inclusive).

4.6.2.4 process_state_t state

Process state (Ready, Running, or Blocked).

4.6.2.5 unsigned char∗ stack_top

Pointer to the top of this processes's stack.

4.6.2.6 unsigned char∗ stack_base

Pointer to the bottom of this processes's stack.

4.6.2.7 int memory_size

Memory size ...

will be used in R3 and R4.

4.6.2.8 unsigned char∗ load_address

Load address ...

will be used in R3 and R4.

4.6.2.9 unsigned char∗ exec_address

Execution address ...

will be used in R3 and R4.

The documentation for this struct was generated from the following file:

- mpx/pcb.h

Index