EPICS Multi-Core Utilities

1.1.2-SNAPSHOT

Generated by Doxygen 1.8.1.2

Mon May 13 2013 14:21:07

Contents

Chapter 1

EPICS Multi-Core Utilities

1.1 Scope of this Document

This documentation covers the C API and the iocShell commands of the EPICS Multi-Core Utilities.

1.2 Introduction

The EPICS Multi-Core Utilities library contains tools that allow tweaking of real-time parameters for EPICS IOC threads running on multi-core processors under the Linux operating system.

These tools are intended to set up multi-core IOCs for fast controllers, by:

- Confining either parts or the complete EPICS IOC onto a subset of the available cores, allowing hard real-time applications and threads to run on dedicated cores.
- Changing priorities of callback, driver or communication threads with respect to database processing.
- Selecting real-time scheduling policy (FIFO or Round-Robin) for selected threads.
- Locking the IOC process virtual memory into RAM to avoid swapping.

1.2.1 Advanced Thread Show Routines

An extended version of the `epicsThreadShow()` command, showing scheduling policy and CPU affinity in addition to the usual output.

Details can be found in the documentation for module Real-Time threadShow Routines.

1.2.2 Rule Based Real-Time Property Manipulation

A module allowing to specify rules, which consist of a regular expression to match the thread name against, and a set of commands that allow to specify the real-time properties of a thread.

Whenever the EPICS IOC starts a thread, its name is matched against all existing rules, and for matching rules the commands are applied.

Details can be found in the documentation for module Rule-Based Thread Properties.

Warning

The default priorities of the EPICS IOC threads are well-chosen. They have been proven to ensure reliable IOC operation and communication, in many installations, under a variety of circumstances.
Manipulating the real-time properties, especially scheduling policies and priorities, may have unwanted side effects. Use this feature sparingly, and test well.

1.2.3 Memory Locking

A module allowing to lock the IOC process virtual memory into RAM. This makes sure that no swapping occurs, and thus avoids page faults which would introduce latency and lead to indeterministic timing.

Details can be found in the documentation for module Memory Locking.

1.3 Sources

Releases can be found at http://sourceforge.net/projects/epics/files/mcoreutils/

The sources are versioned using Mercurial. They can be viewed at http://epics.hg.sourceforge.net/hgweb/epics/mcoreutils/ or checked out using

```
hg clone http://epics.hg.sourceforge.net:8000/hgroot/epics/mcoreutils
```

1.4 Requirements

- Linux operating system
- EPICS BASE 3.15 trunk

1.5 Installation

- Unpack the distribution tar or check out the source tree.
- Run `make`
- To generate a minimal example IOC, run `make -C example`

1.6 Usage

To use the Multi-Core Utilities in an IOC application tree, you have to add a definition to `.../configure/RELEASE` that points to the location of the `mcoreutils` module.

In the directory that builds your IOC binary, the `Makefile` has to make sure the IOC is only built for Linux. Then add the dbd file and the Library, e.g.:

```
...
PROD_IOC_Linux = mcutest
...
mcutest_DBD += mcoreutils.dbd
...
mcutest_LIBS += mcoreutils
...
```

That's it. Enjoy!

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 Real-Time threadShow Routines

Add two new threadShow functions that show scheduling policy and CPU affinity.

Files

- file threadShow.c

  New threadShow showing real-time properties.

Functions

- epicsShareFunc void mcoreThreadShowInit (void)

  Initialization routine.
- epicsShareFunc void mcoreThreadShow (epicsThreadId thread, unsigned int level)

  iocShell: *Show thread info for one thread.*
- epicsShareFunc void mcoreThreadShowAll (unsigned int level)

  iocShell: *Show thread info for all threads.*

4.1.1 Detailed Description

Add two new threadShow functions that show scheduling policy and CPU affinity. Adds two new threadShow functions that, in addition to the properties shown by `epicsThreadShow()` and `epicsThreadShowAll()`, print the scheduling policy, and the CPU affinity of each thread.

Uses the `epicsThreadMap()` call to have a hook function being called for every thread, which prints out the thread properties.

4.1.2 Function Documentation

4.1.2.1 epicsShareFunc void mcoreThreadShow (epicsThreadId *thread,* unsigned int *level*)

iocShell: Show thread info for one thread.

Sets the global thread and level variables, and calls the map function.

Parameters

thread	id of thread to show
level	verbosity level

IOC Shell

mcoreThreadShow thread level

thread	thread name or id
level	verbosity level

Definition at line 122 of file threadShow.c.

4.1.2.2 epicsShareFunc void mcoreThreadShowAll (unsigned int *level*)

iocShell: Show thread info for all threads.

Parameters

level	verbosity level

IOC Shell

mcoreThreadShowAll level

level	verbosity level

Definition at line 136 of file threadShow.c.

4.1.2.3 epicsShareFunc void mcoreThreadShowInit (void)

Initialization routine.

Must be called before using any of the other functions, which is done when registering the iocsh commands.

Definition at line 154 of file threadShow.c.

4.2 Rule-Based Thread Properties

Allow user-specified rules that modify real-time properties of EPICS threads.

Files

- file threadRules.c

  Rule-based modification of thread real-time properties.

Functions

- epicsShareFunc void mcoreThreadModify (epicsThreadId id, const char *policy, const char *priority, const char *cpus)

  iocShell: *Modify a thread's real-time properties.*
- epicsShareFunc void mcoreThreadRulesInit ()

  Initialization routine.
- epicsShareFunc long mcoreThreadRuleAdd (const char *name, const char *policy, const char *priority, const char *cpus, const char *pattern)

  iocShell: *Add or replace a thread rule.*
- epicsShareFunc void mcoreThreadRuleDelete (const char *name)

  iocShell: *Delete a thread rule.*
- epicsShareFunc void mcoreThreadRulesShow (void)

  iocShell: *Print a comprehensive list of the thread rules.*

4.2.1 Detailed Description

Allow user-specified rules that modify real-time properties of EPICS threads. Implements a library that uses rules to modify real-time properties of EPICS threads:

- *Scheduling policy*

  Scheduling mechanism used for this thread. When POSIX scheduling is enabled, the default mechanism is `SCHED_FIFO`, but `SCHED_OTHER` and `SCHED_RR` are also supported.

- *Scheduling priority*

  OSI priority value that gets converted to the system's real-time priority schema.

- *CPU Affinity*

  Set of CPUs that this thread is allowed to run on.

This is achieved by creating a linked list of rules, which consist of a regular expression pattern and modification instructions. A hook function is added to the EPICS thread creation module. The hook is called from every thread as part of its creation, matches the regular expression patterns of all rules against the name of the newly created thread, and applies the modifications of all rules that match.

See Also

See man pages for `pthread_setschedparam(3)` and `sched_setscheduler(2)` for details on scheduling policy and priority, `pthread_setaffinity_np(3)` and `sched_setaffinity(2)` for details on CPU affinity.

Configuration Files

The module tries to read a system configuration file (`/etc/rtrules`) and a user configuration file (default: `$HOME/.rtrules`) to create the initial list of thread rules.

The file format is based on the format of the `/etc/rtgroups` file on RHEL-MRG. Each line has the format

`name:policy:priority:affinity:pattern`

`name`	name of the rule
`policy`	scheduling policy to set for the thread (first letter, not case sensitive), ∗ = don't change
`priority`	scheduling priority to set for the thread (a + or − sign adds to the current priority), ∗ = don't change
`affinity`	CPUs to set the thread's affinity to (use , and − to specify multiple CPUs and ranges, e.g. `0`,3-5), ∗ = don't change
`pattern`	regular expression pattern to match thread names against, see man page for `regex(7)` for details

Lines starting with # (comments), and empty lines (containing only whitespace) are ignored.

Environment Variables

`HOME` location of the `HOME` directory (default: `/`)

`EPICS_MCORE_USERCONFIG` name of user configuration file, relative to the `HOME` directory (default: `.rtrules`)

Known Issues

A thread calling `epicsThreadSetPriority()` to set its priority while running may override the priorities defined in the rules at any time.

4.2.2 Function Documentation

4.2.2.1 epicsShareFunc void mcoreThreadModify (epicsThreadId *id*, const char ∗ *policy*, const char ∗ *priority*, const char ∗ *cpus*)

iocShell: Modify a thread's real-time properties.

Parameters

id	EPICS thread id
policy	scheduling policy to set (∗ = don't change)
priority	scheduling priority (OSI) to set (a + or − sign adds to the current priority, ∗ = don't change)
cpus	cpuset specification to set (use , and − to specify multiple CPUs and ranges, ∗ = don't change)

IOC Shell

`mcoreThreadModify thread policy priority cpus`

`thread`	thread name or id
`policy`	scheduling policy to set (∗ = don't change)
`priority`	scheduling priority (OSI) to set (a + or − sign adds to the current priority, ∗ = don't change)
`cpus`	cpuset specification to set (use , and − to specify multiple CPUs and ranges, ∗ = don't change)

Definition at line 289 of file threadRules.c.

4.2.2.2 epicsShareFunc long mcoreThreadRuleAdd (const char * *name*, const char * *policy*, const char * *priority*, const char * *cpus*, const char * *pattern*)

iocShell: Add or replace a thread rule.

Parameters

name	rule name (identifier)
policy	scheduling policy to set (* = don't change)
priority	scheduling priority (OSI) to set (a + or − sign adds to the current priority, * = don't change)
cpus	cpuset specification to set (use , and − to specify multiple CPUs and ranges, * = don't change)
pattern	`regex(7)` pattern to match thread names against

Returns

(OK, ERROR) as (0,-1)

IOC Shell

`mcoreThreadRuleAdd name policy priority cpus pattern`

`name`	rule name (identifier)
`policy`	scheduling policy to set (* = don't change)
`priority`	scheduling priority (OSI) to set (a + or − sign adds to the current priority, * = don't change)
`cpus`	cpuset specification to set (use , and − to specify multiple CPUs and ranges, * = don't change)
`pattern`	`regex(7)` pattern to match thread names against

Definition at line 109 of file threadRules.c.

4.2.2.3 epicsShareFunc void mcoreThreadRuleDelete (const char * *name*)

iocShell: Delete a thread rule.

Parameters

name	name (identifier) of the rule to delete

IOC Shell

`mcoreThreadRuleDelete name`

`name`	name (identifier) of the rule to delete

Definition at line 140 of file threadRules.c.

4.2.2.4 epicsShareFunc void mcoreThreadRulesInit ( )

Initialization routine.

Must be called before using any of the other functions, which is done when registering the iocsh commands.

Definition at line 377 of file threadRules.c.

4.2.2.5 epicsShareFunc void mcoreThreadRulesShow (void)

iocShell: Print a comprehensive list of the thread rules.

Rule names are shortened to 16 characters.

IOC Shell

```
mcoreThreadRulesShow
```

Definition at line 165 of file threadRules.c.

4.3 Memory Locking

Add functions for locking the process memory into RAM.

Files

- file memLock.c

  Locking process memory into RAM.

Functions

- epicsShareFunc void mcoreMLock (void)

  iocShell: *Lock all process virtual memory into RAM.*
- epicsShareFunc void mcoreMUnlock (void)

  iocShell: *Unlock process virtual memory from RAM.*

4.3.1 Detailed Description

Add functions for locking the process memory into RAM. Adds functions that allow locking and unlocking the process virtual memory into RAM to make sure no page faults occur, which would introduce unpredictable interruptions and latency.

See Also

See man page for `mlockall(2)` for more details on memory locking.

4.3.2 Function Documentation

4.3.2.1 epicsShareFunc void mcoreMLock (void)

iocShell: Lock all process virtual memory into RAM.

IOC Shell

```
mcoreMLock
```

Definition at line 32 of file memLock.c.

4.3.2.2 epicsShareFunc void mcoreMUnlock (void)

iocShell: Unlock process virtual memory from RAM.

IOC Shell

```
mcoreMUnlock
```

Definition at line 38 of file memLock.c.

Chapter 5

File Documentation

5.1 mcoreutils.h File Reference

```
#include <unistd.h>
#include <epicsThread.h>
#include <shareLib.h>
```

Functions

- epicsShareFunc void mcoreThreadShowInit (void)

  Initialization routine.
- epicsShareFunc void mcoreThreadShow (epicsThreadId thread, unsigned int level)

  iocShell: *Show thread info for one thread.*
- epicsShareFunc void mcoreThreadShowAll (unsigned int level)

  iocShell: *Show thread info for all threads.*
- epicsShareFunc void mcoreThreadModify (epicsThreadId id, const char ∗policy, const char ∗priority, const char ∗cpus)

  iocShell: *Modify a thread's real-time properties.*
- epicsShareFunc void mcoreThreadRulesInit ()

  Initialization routine.
- epicsShareFunc long mcoreThreadRuleAdd (const char ∗name, const char ∗policy, const char ∗priority, const char ∗cpus, const char ∗pattern)

  iocShell: *Add or replace a thread rule.*
- epicsShareFunc void mcoreThreadRuleDelete (const char ∗name)

  iocShell: *Delete a thread rule.*
- epicsShareFunc void mcoreThreadRulesShow (void)

  iocShell: *Print a comprehensive list of the thread rules.*
- epicsShareFunc void mcoreMLock (void)

  iocShell: *Lock all process virtual memory into RAM.*
- epicsShareFunc void mcoreMUnlock (void)

  iocShell: *Unlock process virtual memory from RAM.*

5.1.1 Detailed Description

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file mcoreutils.h.

5.2 memLock.c File Reference

Locking process memory into RAM.

```
#include <stdio.h>
#include <sys/mman.h>
#include <errlog.h>
#include <shareLib.h>
#include "mcoreutils.h"
```

Functions

- void mcoreMLock (void)

  iocShell: *Lock all process virtual memory into RAM.*
- void mcoreMUnlock (void)

  iocShell: *Unlock process virtual memory from RAM.*

5.2.1 Detailed Description

Locking process memory into RAM.

Author

Ralph Lange Ralph.Lange@gmx.de
Dirk Zimoch Dirk.Zimoch@psi.ch

Copyright

Definition in file memLock.c.

5.3 shellCommands.c File Reference

iocShell registration of MCoreUtils commands.

```
#include <unistd.h>
#include <stdlib.h>
#include <iocsh.h>
#include <epicsExport.h>
#include <epicsThread.h>
#include "mcoreutils.h"
```

5.3.1 Detailed Description

iocShell registration of MCoreUtils commands.

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file shellCommands.c.

5.4 threadRules.c File Reference

Rule-based modification of thread real-time properties.

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <regex.h>
#include <string.h>
#include <ellLib.h>
#include <envDefs.h>
#include <errlog.h>
#include <epicsStdio.h>
#include <epicsMath.h>
#include <epicsThread.h>
#include <epicsMutex.h>
#include <shareLib.h>
#include "utils.h"
#include "mcoreutils.h"
```

- typedef struct threadRule threadRule

  A thread rule.

- long mcoreThreadRuleAdd (const char *name, const char *policy, const char *priority, const char *cpus, const char *pattern)

  Add or replace a thread rule.

- void mcoreThreadRuleDelete (const char *name)

  Delete a thread rule.

- void mcoreThreadRulesShow (void)

  Print a comprehensive list of the thread rules.

- void mcoreThreadModify (epicsThreadId id, const char *policy, const char *priority, const char *cpus)

  Modify a thread's real-time properties.

- void mcoreThreadRulesInit (void)

  Initialization routine.

5.4.1 Detailed Description

Rule-based modification of thread real-time properties.

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file threadRules.c.

5.4.2 Typedef Documentation

5.4.2.1 typedef struct threadRule threadRule

A thread rule.

Used to manipulate real-time properties when threads are started. The thread rules are kept in a linked list.

5.5 threadShow.c File Reference

New threadShow showing real-time properties.

```
#include <stdlib.h>
#include <sched.h>
#include <string.h>
#include <pthread.h>
#include <ellLib.h>
#include <errlog.h>
#include <epicsStdio.h>
#include <epicsEvent.h>
#include <epicsThread.h>
#include <epicsMath.h>
#include <shareLib.h>
#include "utils.h"
#include "mcoreutils.h"
```

- void mcoreThreadShow (epicsThreadId thread, unsigned int level)

  Show thread info for one thread.
- void mcoreThreadShowAll (unsigned int level)

  Show thread info for all threads.
- void mcoreThreadShowInit (void)

  Initialization routine.

5.5.1 Detailed Description

New threadShow showing real-time properties.

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file threadShow.c.

5.6 utils.c File Reference

Utility functions for MCoreUtils.

```
#include <stdlib.h>
#include <stdio.h>
#include <sched.h>
#include <string.h>
#include <errlog.h>
#include "utils.h"
```

Functions

- void strToCpuset (cpu_set_t *cpuset, const char *spec)
  Convert a cpuset string specification (e.g. "0,2-3") to a cpuset.
- void cpusetToStr (char *set, size_t len, const cpu_set_t *cpuset)
  Convert a cpuset into its string specification (e.g. "0,2-3").
- const char * policyToStr (const int policy)
  Convert scheduling policy to string.
- int strToPolicy (const char *string)
  Convert string policy specification to policy.

Variables

- epicsShareDef int cpuDigits
  Number of digits needed for a single CPU spec.

5.6.1 Detailed Description

Utility functions for MCoreUtils.

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file utils.c.

5.6.2 Function Documentation

5.6.2.1 void cpusetToStr (char * *set*, size_t *len*, const cpu_set_t * *cpuset*)

Convert a cpuset into its string specification (e.g. "0,2-3").

Parameters

set	output buffer to write into
len	length of `set`
cpuset	cpuset to convert

Definition at line 59 of file utils.c.

5.6.2.2 const char* policyToStr (const int *policy*)

Convert scheduling policy to string.

Parameters

policy	policy to convert

Returns

string representation

Definition at line 96 of file utils.c.

5.6.2.3 void strToCpuset (cpu_set_t * *cpuset*, const char * *spec*)

Convert a cpuset string specification (e.g. "0,2-3") to a cpuset.

Parameters

cpuset	cpuset to write into
spec	specification string

Definition at line 29 of file utils.c.

5.6.2.4 int strToPolicy (const char * *string*)

Convert string policy specification to policy.

Parameters

string	string policy specification

Returns

policy value, or -1 on error

Definition at line 124 of file utils.c.

5.6.3 Variable Documentation

5.6.3.1 epicsShareDef int cpuDigits

Number of digits needed for a single CPU spec.

Set in mcoreThreadShowInit().

Definition at line 21 of file utils.c.

5.7 utils.h File Reference

Header file for utils.c.

```
#include <sched.h>
#include <unistd.h>
#include <errlog.h>
```

Macros

- #define NO_OF_CPUS sysconf(_SC_NPROCESSORS_CONF)
- #define checkStatus(status, message)

Functions

- void strToCpuset (cpu_set_t *cpuset, const char *spec)

  Convert a cpuset string specification (e.g. "0,2-3") to a cpuset.
- void cpusetToStr (char *set, size_t len, const cpu_set_t *cpuset)

  Convert a cpuset into its string specification (e.g. "0,2-3").
- const char * policyToStr (const int policy)

  Convert scheduling policy to string.
- int strToPolicy (const char *string)

  Convert string policy specification to policy.

Variables

- int cpuDigits

  Number of digits needed for a single CPU spec.

5.7.1 Detailed Description

Header file for utils.c.

Author

Ralph Lange Ralph.Lange@gmx.de

Copyright

Definition in file utils.h.

5.7.2 Macro Definition Documentation

5.7.2.1 #define checkStatus(*status, message*)

Value:

```
if((status))  {\
    errlogPrintf("%s error %s\n", (message), strerror((status))); \
}
```

Definition at line 23 of file utils.h.

5.7.2.2 #define NO_OF_CPUS sysconf(_SC_NPROCESSORS_CONF)

Definition at line 21 of file utils.h.

5.7.3 Function Documentation

5.7.3.1 void cpusetToStr (char * *set,* size_t *len,* const cpu_set_t * *cpuset*)

Convert a cpuset into its string specification (e.g. "0,2-3").

Parameters

set	output buffer to write into
len	length of `set`
cpuset	cpuset to convert

Definition at line 59 of file utils.c.

5.7.3.2 const char* policyToStr (const int *policy*)

Convert scheduling policy to string.

Parameters

policy	policy to convert

Returns

string representation

Definition at line 96 of file utils.c.

5.7.3.3 void strToCpuset (cpu_set_t * *cpuset,* const char * *spec*)

Convert a cpuset string specification (e.g. "0,2-3") to a cpuset.

Parameters

cpuset	cpuset to write into
spec	specification string

Definition at line 29 of file utils.c.

5.7.3.4 int strToPolicy (const char * *string*)

Convert string policy specification to policy.

Parameters

string	string policy specification

Returns

policy value, or -1 on error

Definition at line 124 of file utils.c.

5.7.4 Variable Documentation

5.7.4.1 int cpuDigits

Number of digits needed for a single CPU spec.

Set in mcoreThreadShowInit().

Definition at line 21 of file utils.c.

Index