무료 전자 책

배우기

linux-kernel

Free unaffiliated eBook created from
Stack Overflow contributors.

#linux-kernel

1
1: linux-kernel 2
2
2
Examples 2
2
. 2
, . 2
2: Linux : (FIFO) 4
Examples 4
Named Pipe (FIFO) ? 4
3: Linux Hello World 5
Examples 5
5
5
4: . 7
7
Examples 7
FTDI USB "" 7
5: 8
Examples 8
I2C 8
6: 9
9
Examples 9
9
7: 11
Examples 11
fork () 11
13

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: linux-kernel

It is an unofficial and free linux-kernel ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official linux-kernel.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: linux-kernel

linux-kernel , .

linux-kernel . linux-kernel .

4.4	2016-01-10
4.1	2015-06-21
3.18	2014-12-07
3.16	2014-08-03
3.12	2013-11-03
3.10	2013-06-30
3.4	2012-05-20
3.2	2012-01-04

Examples

Linux https://www.kernel.org/ .

.

(linux-4.7.tar.gz)

```
wget http://www.kernel.org/pub/linux/kernel/v4.7/linux-4.7.tar.gz
tar zxvf linux-4.7.tar.gz
cd linux-4.7
```

`make menuconfig` . `.config` `make oldconfig` . `make xconfig` .

, .

```
make dep
make bzImage
make modules
make modules_install
```

.

```
make mrproper
make menuconfig
make dep
make clean
make bzImage
make modules
make modules_install
```

```
system.map /boot/vmlinuz-4.7 .
```

```
.conf .
```

```
image = /boot/vmlinuz-4.7
label = "Linux 4.7"
```

```
lilo -v    .
```

linux-kernel : https://riptutorial.com/ko/linux-kernel/topic/2385/linux-kernel-

2: Linux : (FIFO)

Examples

Named Pipe (FIFO) ?

```
A named pipe is really just a special kind of file (a FIFO file) on the local hard drive.
Unlike a regular file, a FIFO file does not contain any user information. Instead, it allows
two or more processes to communicate with each other by reading/writing to/from this file.

A named pipe works much like a regular pipe, but does have some noticeable differences.

Named pipes exist as a device special file in the file system.
Processes of different ancestry can share data through a named pipe.
When all I/O is done by sharing processes, the named pipe remains in the file system for later
use.

The easiest way to create a FIFO file is to use the mkfifo command. This command is part of
the standard Linux utilities and can simply be typed at the command prompt of your shell. You
may also use the mknod command to accomplish the same thing.
```

Linux : (FIFO) : https://riptutorial.com/ko/linux-kernel/topic/6144/linux-------fifo-

3: Linux Hello World

Examples

```
#include <linux/init.h>
#include <linux/module.h>

/**
 * This function is called when the module is first loaded.
 */
static int __init hello_kernel_init(void)
{
    printk("Hello, World!\n");
    return 0;
}

/**
 * This function is called when is called if and when the module is unloaded.
 */
static void __exit hello_kernel_exit(void)
{
    printk("Goodbye, cruel world...\n");
}

/* The names of the init/exit functions are arbitrary, and they are bound using the following
macro definitions */
module_init(hello_kernel_init);
module_exit(hello_kernel_exit);
```

Linux (Character-device, Block-device) .

. . - , .

Linux .

`/lib/modules/<kernel-version>` Makefile `driver.c driver.ko`

```
obj-m := driver.o
KDIR := /lib/modules/$(shell uname -r)/build/
PWD := $(shell pwd)

all:
    $(MAKE) -C $(KDIR) M=$(PWD) modules
```

Makefile `make` .

src :

```
driver.c  driver.ko  driver.mod.c  driver.mod.o  driver.o  Makefile  modules.order
Module.symvers
```

"", :

```
$ insmod driver.ko
$ dmesg | tail -n 1
[133790.762185] Hello, World!

$ rmmod driver.ko
$ dmesg | tail -n 1
[133790.762185] Goodbye, cruel world...
```

Linux Hello World : https://riptutorial.com/ko/linux-kernel/topic/7056/linux-hello-world--

4: .

tovalds/linux .MAINTAINERS , linux-kernel , git log , scripts/get-maintainer .

Examples

FTDI USB "" .

. drivers/usb/serial/ftdi_sio.c .

```
./scripts/get_maintainer.pl drivers/usb/serial/ftdi_sio.c
```

:

```
Johan Hovold <johan@kernel.org> (maintainer:USB SERIAL SUBSYSTEM)
Greg Kroah-Hartman <gregkh@linuxfoundation.org> (supporter:USB SUBSYSTEM)
linux-usb@vger.kernel.org (open list:USB SERIAL SUBSYSTEM)
linux-kernel@vger.kernel.org (open list)
```

.

. : https://riptutorial.com/ko/linux-kernel/topic/10056/----

5:

Examples

I2C

: `debugfs /sys/kernel/debug .`

.

```
mount -t debugfs none /sys/kernel/debug
```

.

```
cd /sys/kernel/debug/tracing/
```

.

```
echo nop > current_tracer
```

I2C :

```
echo 1 > events/i2c/enable
```

.

```
echo 1 > tracing_on
```

`/sys/kernel/debug/tracing/trace` (:

```
... i2c_write: i2c-5 #0 a=044 f=0000 l=2 [02-14]
... i2c_read: i2c-5 #1 a=044 f=0001 l=4
... i2c_reply: i2c-5 #1 a=044 f=0001 l=4 [33-00-00-00]
... i2c_result: i2c-5 n=2 ret=2
```

API `Documentation/trace/events.txt` .

: https://riptutorial.com/ko/linux-kernel/topic/3466/-

6:

.

Examples

kern_thread.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/kthread.h>
#include <linux/sched.h>

#define AUTHOR        "Nachiket Kulkarni"
#define DESCRIPTION    "Simple module that demonstrates creation of 2 kernel threads"

static int kthread_func(void *arg)
{
/* Every kthread has a struct task_struct associated with it which is it's identifier.
* Whenever a thread is schedule for execution, the kernel sets "current" pointer to
* it's struct task_struct.
* current->comm is the name of the command that caused creation of this thread
* current->pid is the process of currently executing thread
*/
    printk(KERN_INFO "I am thread: %s[PID = %d]\n", current->comm, current->pid);
    return 0;
}

static int __init init_func(void)
{
    struct task_struct *ts1;
    struct task_struct *ts2;
    int err;

    printk(KERN_INFO "Starting 2 threads\n");

/*struct task_struct *kthread_create(int (*threadfn)(void *data), void *data, \
 *                          const char *namefmt, ...);
 * This function creates a kernel thread and starts the thread.
 */
    ts1 = kthread_run(kthread_func, NULL, "thread-1");
    if (IS_ERR(ts1)) {
        printk(KERN_INFO "ERROR: Cannot create thread ts1\n");
        err = PTR_ERR(ts1);
        ts1 = NULL;
        return err;
    }

    ts1 = kthread_run(kthread_func, NULL, "thread-1");
    if (IS_ERR(ts1)) {
        printk(KERN_INFO "ERROR: Cannot create thread ts1\n");
        err = PTR_ERR(ts1);
        ts1 = NULL;
        return err;
    }
```

```
    printk(KERN_INFO "I am thread: %s[PID = %d]\n", current->comm, current->pid);
    return 0;
}

static void __exit exit_func(void)
{
    printk(KERN_INFO "Exiting the module\n");
}

module_init(init_func);
module_exit(exit_func);

MODULE_AUTHOR(AUTHOR);
MODULE_DESCRIPTION(MODULE_AUTHOR);
MODULE_LICENSE("GPL");
```

Makefile

```
obj-m += kern_thread.o

all:
    make -C /lib/module/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/module/$(shell uname -r)/build M=$(PWD) clean
```

.ko .

```
Starting 2 threads
I am thread: thread-1[PID = 6786]
I am thread: insmod[PID = 6785]
I am thread: thread-2[PID = 6788]
```

: https://riptutorial.com/ko/linux-kernel/topic/10619/----

7:

Examples

fork ()

`fork()` . fork (`fork()`) . . , .

PID (ID). PPID (ID) PID .

:

: `#include <unistd.h>`
: `pid_t fork(void);`

fork () .

PID 0 . `-1` .

:

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

void child_process();
void parent_process();

int main()
{
    pid_t pid;
    pid=fork();
    if(pid==0)
        child_process();
    else
        parent_process();
    return 0;
}

/*getpid() will return the Pid of the
  current process executing the function */

void child_process()
{
    printf("Child process with PID : %d  and PPID : %d ", getpid(),getppid());
}

void parent_process()
{
    printf("Parent process with PID : %d", getpid());
}
```

`printf` .

: https://riptutorial.com/ko/linux-kernel/topic/5199/--

S. No		Contributors
1	linux-kernel	Community, EsmaeelE, Marek Skiba, Matt, Tejus Prasad, vinay hunachyal
2	Linux : (FIFO)	chait
3	Linux Hello World	Gilad Naaman
4	.	DevNull, EsmaeelE
5		sergej
6		nachiketkulk
7		chait, Dilip Kumar