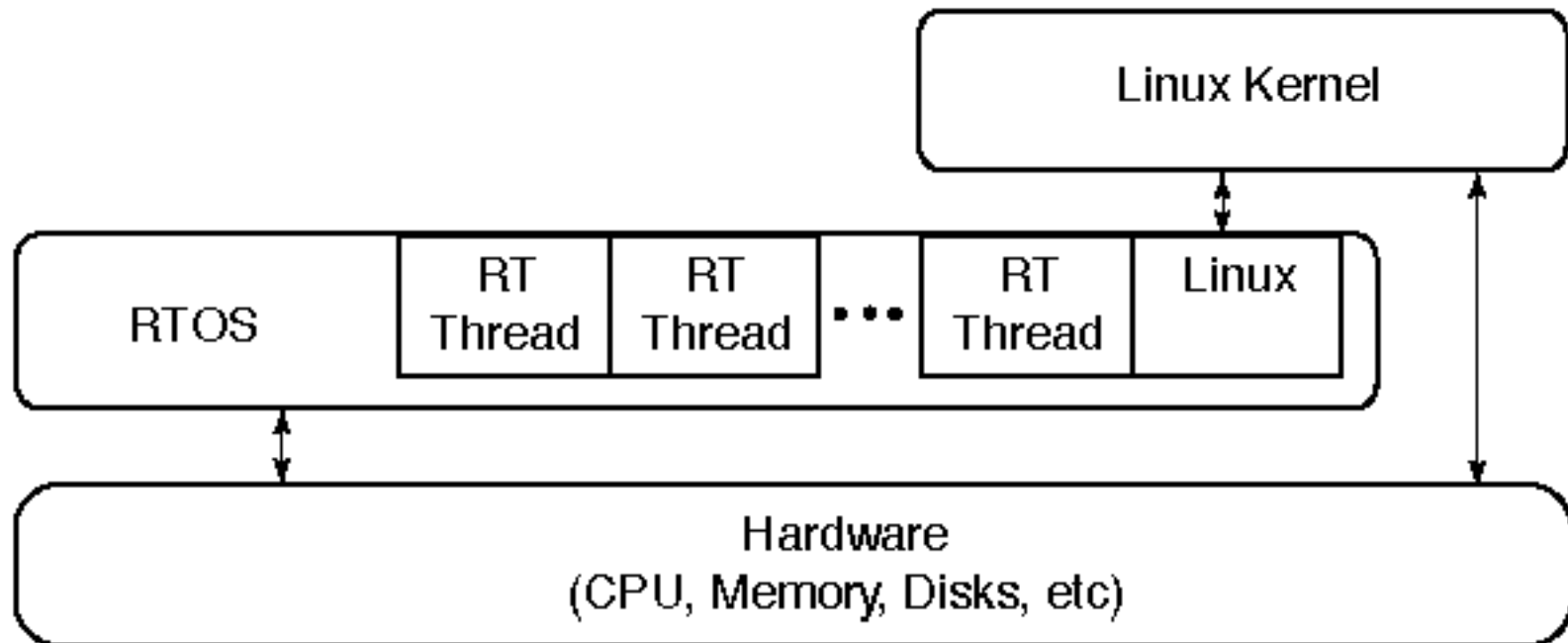


Real-Time Linux (RTLinux) Intro



- Linux with RT extensions (RTOS w/ Linux)

RTLinux “Programs”

- RTLinux programs run as kernel modules
- No virtual memory protection in kernel space
- Many library routines not available
- RT threads cannot call many kernel functions

Boot to RTLinux kernel

- Multi-lab machines have multiple kernels
- Press shift key *before* LILO prompt
- Select EE599_RTLinux at LILO prompt
- `uname -r` tells active kernel release name

```
aster% uname -r  
2.4.18-rt13.2-pre1
```

Module Basics

- Modules can be installed without rebooting
- Each module has:
 - ◇ Initialization Routine
 - ◇ Cleanup Routine
 - ◇ Other code

Simple Module: hello.c

```
#define __KERNEL__
#define MODULE

#include <linux/modversions.h>
#include <linux/module.h>

int init_module(void)
{
    printk("Hello, world\n");
    return 0;
}

void cleanup_module(void)
{
    printk("Goodbye, world\n");
}
```

Compiling and Using Kernel Modules

```
aster% gcc -Wall -c hello.c
hello.c: In function `init_module':
hello.c:9: warning: implicit
declaration of function `printk'
aster% sudo /sbin/insmod hello.o
Hello, World
aster% /sbin/lsmod
Module              Size      Used by
hello                368        0 (unused)
vfat                 9200       0 (unused)
msdos                4720       0 (unused)
fat                 29024      0 [vfat msdos]
tulip                37392      1
aster% sudo /sbin/rmmod hello
Goodbye, world
```

Always remove your modules when done

Real-Time Modules (Declarations)

```
#include <rtl.h>  
#include <time.h>  
#include <rtl_sched.h>
```

```
pthread_t tasks[2];
```

```
void *hello_thr(void *arg);  
void *world_thr(void *arg);
```

RTLinux Modules (Init & Cleanup)

```
int init_module(void)
{
    pthread_create(&tasks[0], NULL,
                  hello_thr, NULL);
    pthread_create(&tasks[1], NULL,
                  world_thr, NULL);
    return 0;
}

void cleanup_module(void)
{
    pthread_cancel(tasks[0]);
    pthread_join(tasks[0], NULL);
    pthread_cancel(tasks[1]);
    pthread_join(tasks[1], NULL);
}
```

RTLinux Modules (RT Threads)

```
void *hello_thr(void *arg)
{
    pthread_make_periodic_np(
        pthread_self(), gethrtime(),
        500000000);

    while(1) {
        pthread_wait_np();
        rtl_printf("hello\n");
    }
    return 0;
}

/* world_thr is similar */
```

RTLlinux Modules (Compilation)

```
aster% cat Makefile
```

```
include /homes/dieter/ee599/rtlinux/rtl.mk
```

```
rthello.o: rthello.c
```

```
aster% make
```

```
gcc -D__KERNEL__ -Wall -Wstrict-prototypes -Wno-trigraphs -fno-strict-aliasing -  
fno-common -pipe -mpreferred-stack-boundary=2 -march=k6 -DMODULE -g -D__RTL__ -  
D_LOOSE_KERNEL_NAMES -O2 -I/u/al-d7/csguest/dieter/ee599/rtlinux/linux/include -  
I/a/al/u/al-d7/csguest/dieter/ee599/rtlinux-3.2-pre1/include -I/a/al/u/al-  
d7/csguest/dieter/ee599/rtlinux-3.2-pre1/include/compat -I/a/al/u/al-  
d7/csguest/dieter/ee599/rtlinux-3.2-pre1/include/posix -c -o rthello.o rthello.c
```

```
#!/homes/dieter/ee599/rtlinux/scripts
```

```
should be in $PATH
```

```
aster% rtlinux start rthello.o
```

```
...
```

```
aster% rtlinux stop rthello
```

Other Useful RTLinux Features

- Waiting a set amount of time

```
int nanosleep(req, remain);  
int clock_nanosleep(clk, flags, req,  
                    remain);
```

- FIFOs

```
int rtf_create(fifo, size);  
int rtf_destroy(fifo);  
int rtf_get(fifo, buf, count);  
int rtf_put(fifo, buf, count);
```

More Info on RTLinux

- Manual pages, white papers, example code
- “Useful Links” section of EE 599 web page