

华中科技大学

课程实验报告

课程名称： 嵌入式操作系统

专业班级：

学 号：

姓 名：

指导教师：

报告日期：

计算机科学与技术学院

目录

实验一 进程控制	3
1 实验目的与要求	3
2 实验内容	3
3 实验过程与结果	3
4 实验结果分析	4
5 心得体会与总结	5
实验二 线程同步与通信	6
1 实验目的与要求	6
2 实验内容	6
3 实验过程与结果	6
4 实验结果分析	7
5 心得体会与总结	7
实验三 共享内存与进程同步	9
1 实验目的与要求	9
2 实验内容	9
3 实验过程与结果	9
4 实验结果分析	10
5 心得体会与总结	10
实验四 TinyOS 实验	12
1 实验目的与要求	12
2 实验内容	12
3 实验过程与结果	13
4 实验结果分析	19
5 心得体会与总结	20
附加实验 Linux 文件目录	21
1 实验目的与要求	21
2 实验内容	21
3 实验过程与结果	21
4 实验结果分析	23
5 心得体会与总结	25
附录	26

实验一 进程控制

1 实验目的与要求

1. 加深对进程的理解,进一步认识并发执行的实质;
2. 分析进程争用资源现象,学习解决进程互斥的方法;
3. 掌握 Linux 进程基本控制;
4. 掌握 Linux 系统中的软中断和管道通信。

2 实验内容

编写程序,演示多进程并发执行和进程软中断、管道通信。

- a) 父进程使用系统调用 `pipe()` 建立一个管道,然后使用系统调用 `fork()` 创建两个子进程,子进程 1 和子进程 2;

- b) 子进程 1 每隔 1 秒通过管道向子进程 2 发送数据:

I send you x times. (x 初值为 1, 每次发送后做加一操作)

子进程 2 从管道读出信息,并显示在屏幕上。

- c) 父进程用系统调用 `signal()` 捕捉来自键盘的中断信号(即按 `Ctrl+C` 键);当捕捉到中断信号后,父进程用系统调用 `kill()` 向两个子进程发出信号,子进程捕捉到信号后分别输出下列信息后终止:

Child Process 1 is Killed by Parent!

Child Process 2 is Killed by Parent!

- d) 父进程等待两个子进程终止后,释放管道并输出如下的信息后终止

Parent Process is Killed!

3 实验过程与结果

1. 调用 `pipe` 函数创建一个无名管道,使两个进程通过这个管道来互相传递数据。
2. 调用两次 `fork` 函数创建两个子进程。
3. 主进程创建结束子进程后,调用 `signal` 函数,监听 `SIGINT` 键盘中断信号,并设置为监听到则执行 `handle` 函数来完成指定的结束两个子进程的任务,然后调用 `wait` 函数等待两个子进程运行结束。
4. 一号子进程进入其主循环(一个死循环),不停地向管道中写入数据。
5. 二号子进程进入其主循环(一个死循环),不停从管道中读取数据,并输出到控制台中。
6. 当控制台中输入 `SIGINT` 中断信号后,主进程结束了两个子进程,关闭管道,结束程序。

7. 运行结果:

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$ ./lab1.out
-----successfully-----
I send you 0 times.
I send you 1 times.
I send you 2 times.
I send you 3 times.
I send you 4 times.
I send you 5 times.
I send you 6 times.
I send you 7 times.
```

图 1.1 程序运行截图

```
I send you 13 times.
I send you 14 times.
I send you 15 times.
I send you 16 times.
I send you 17 times.
I send you 18 times.
^CChild Process 1 is Killed by Parent!
Child Process 2 is Killed by Parent!
Parent Process is Killed!
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$
```

图 1.2 自我结束进程截图

4 实验结果分析

1. 编译源程序:

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$ gcc lab1.c -o lab1.out
```

图 1.3 gcc 编译截图

2. 运行过程:

图 1.1, 1.2 展示了程序的整个运行过程, 程序在成功初始化管道、子进程后, 按顺序输出了指定的计次字符串。在控制台中按下 CTRL+C 后, 程序成功监听到了 SIGINT 中断信号, 完成了对子进程的结束。

3. 监听 SIGINT 信号后进行中断:

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$ ps
  PID TTY          TIME CMD
   34 tty2          00:00:00 bash
   94 tty2          00:00:00 ps
```

图 1.4 后台进程截图

通过 ps 命令查看后台进程, 发现没有实验程序在运行, 说明的确成功完成了对进程的结束。

5 心得体会与总结

1. 第一次实验中，有大量的 Linux C 系统库函数是第一次接触，如管道如何创建，使用，读写等等。课程文档上对函数的说明并不是很完整，在搜索了一些较为完整的函数 API 使用说明后，才摸到了门路。
2. 课程文档中本来是想让我们自定义一个结束进程的函数，但是我在代码中使用了 SIGKILL 这个特殊信号来直接完成结束进程的工作。进一步体会到 Linux 中四十多个信号的不同功用。
3. 第一次写 C 语言的并发代码，感觉虽然通过调用操作系统功能的方式来实现进程的各个功能虽然显得底层一些，但是代码上并不繁杂，内部库的设计还是非常好。

实验二 线程同步与通信

1 实验目的与要求

1. 掌握 Linux 下线程的概念；
2. 了解 Linux 线程同步与通信的主要机制；
3. 通过信号灯操作实现线程间的同步与互斥。

2 实验内容

编写程序，演示多进程并发执行和进程软中断、管道通信。

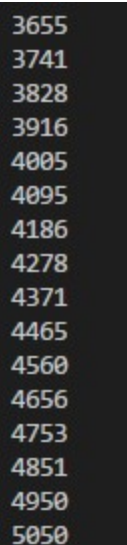
- a) 通过 Linux 多线程与信号灯机制，设计并实现计算机线程与 I/O 线程共享缓冲区的同步与通信。
- b) 程序要求:两个线程,共享公共变量 a，线程 1 负责计算(1 到 100 的累加，每次加一个数)，线程 2 负责打印（输出累加的中间结果）。

3 实验过程与结果

1. 主线程中调用 `semget` 函数创建一个长度为二的信号量数组。
2. 调用 `semctl` 函数来为这个信号量数组完成信号量大小的初始化（第一个初始化为 0，第二个初始化为 1）。
3. 通过 `pthread_create` 函数来创建两个线程，并绑定上指定的运行函数，然后主线程通过 `pthread_join` 函数阻塞自己等待两个线程运行结束。
4. 一号线程进入 100 次的循环体，每次循环时，先对第一个信号量进行 P 操作，再为一个全局变量 `sum` 进行加法操作，在对第二个信号量进行 V 操作。
5. 二号线程进入 100 次的循环体，每次循环时，先对第二个信号量进行 P 操作，再输出全局变量 `sum` 的值，最后对第一个信号量进行 V 操作。
6. 待两个线程运行结束后，主线程删除信号量，结束程序
7. 运行结果：

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$ cd "/mnt/d/Workplace/C/VSCODE/lab2/" && gcc lab2.c -o  
-lpthread && "/mnt/d/Workplace/C/VSCODE/lab2/"lab2.out  
-----hello world!-----  
successfully initialized  
tid: (0x7f99d0a00700)  
threads have been created  
tid: (0x7f99d01f0700)  
1  
3  
6  
10  
15  
21  
28  
36  
45  
55  
66
```

图 2.1 程序运行截图

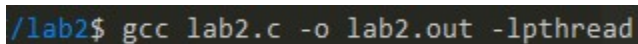


3655
3741
3828
3916
4005
4095
4186
4278
4371
4465
4560
4656
4753
4851
4950
5050

图 2.2 程序的输出最后部分截图

4 实验结果分析

1. 编译源程序:



```
/lab2$ gcc lab2.c -o lab2.out -lpthread
```

图 2.3 gcc 编译截图

由截图可见，成功完成了编译。

2. 运行结果分析:

运行的过程截图如图 2.1, 2.2 所示，可见输出的数列符合实验要求中所描述的 1-100 的累加过程。

5 心得体会与总结

1. 本次实验关注于线程的同步互斥操作，因为有类似的编程经历，完成代码逻辑的过程并不困难，主要的时间还是花在查阅 Linux C 的线程库函数 API 上。
2. 在实验一开始的时候，因为对线程库的功能不是很熟悉，在需要阻塞主线程的时候，不知道可以使用 `pthread_join` 函数来实现，使用了一个全局变量计数器，除非子线程结束的时候更新计数器变量，不然主线程就通过 `busy-wait` 循环的方式进行阻塞，这样实现的方式就不是很优雅。

3. 本程序中的 Linux 中的信号量 P、V 操作是使用了 `semop` 函数传入结构体的方式，感觉这样的传参方式比较复杂，而且 P、V 操作的需求仅仅是判断并自加或自减，这样的需求应该有更简单的实现。

实验三 共享内存与进程同步

1 实验目的与要求

1. 掌握 Linux 下共享内存的概念与使用方法；
2. 掌握环形缓冲的结构与使用方法；
3. 掌握 Linux 下进程同步与通信的主要机制。

2 实验内容

利用多个共享内存(有限空间)构成的环形缓冲,将源文件复制到目标文件,实现两个进程的誊抄。

3 实验过程与结果

1. 主进程中调用 `fopen` 函数打开命令行参数中指定的待读与待写文件。
2. 调用 `shmget` 函数来创建一块指定大小的共享内存空间,在这片空间中,分为两块,主要的部分用来做誊抄的缓冲区,另有一个标志位,存有一个数字,用来标记是否读取完毕待读文件的内容,以及最后一个数据块在缓冲区的索引。
3. 调用 `semget` 函数来创建一个长度为 2 的信号量数组,分别为两个信号量赋值大小为 0 和 128 (即缓冲区的长度),分别代表缓冲区中有多少数据和有多少空位。
4. 主进程调用两次 `fork` 函数创建两个子进程,进行共享内存中的誊抄步骤,然后调用 `wait` 函数阻塞自身,等待两个子进程运行结束。
5. 负责将数据从待读文件复制到缓冲区的线程先调用 `shmat` 函数获得共享内存空间的地址,为标志位初始化值为-1 (即代表本进程尚未完成数据复制)。进入进程的主循环,循环的条件为能通过 `fread` 函数读取到数据,在循环体内对信号量 2 进行 P 操作,再对共享内存空间的指定位置进行复制,移动索引,最后对信号量 1 进行 V 操作。读到最后一个数据的时候,对共享区的索引标志位进行复制。(之所以在互斥区内完成共享索引标志位复制,是为了防止另一个子进程太早读完而则色,没办法得到跳出循环条件,发生死锁)。
6. 负责将数据从缓冲区复制到待写文件的线程先调用 `shmat` 函数获得共享内存空间的地址,然后进入其主循环。对信号量 1 进行 P 操作,从缓冲区获得数据,调用 `fwrite` 函数向待写文件写入数据,判断当前索引是否和标志位中的索引相同,若相同则跳出主循环,最后对信号量 2 进行 V 操作。
7. 誊抄结束后,两个子进程也运行结束,主进程结束阻塞,释放共享内存空间,删除信号量,结束程序。
8. 运行结果:

```

Gray@MICRO50-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab3$ ls
1.pdf lab3.c lab3.out README.md
Gray@MICRO50-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab3$ ./lab3.out 1.pdf 2.pdf
-----start copying!-----
successfully initalized
finish copying
Gray@MICRO50-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab3$ ls
1.pdf 2.pdf lab3.c lab3.out README.md
Gray@MICRO50-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab3$

```

图 3.1 程序运行截图

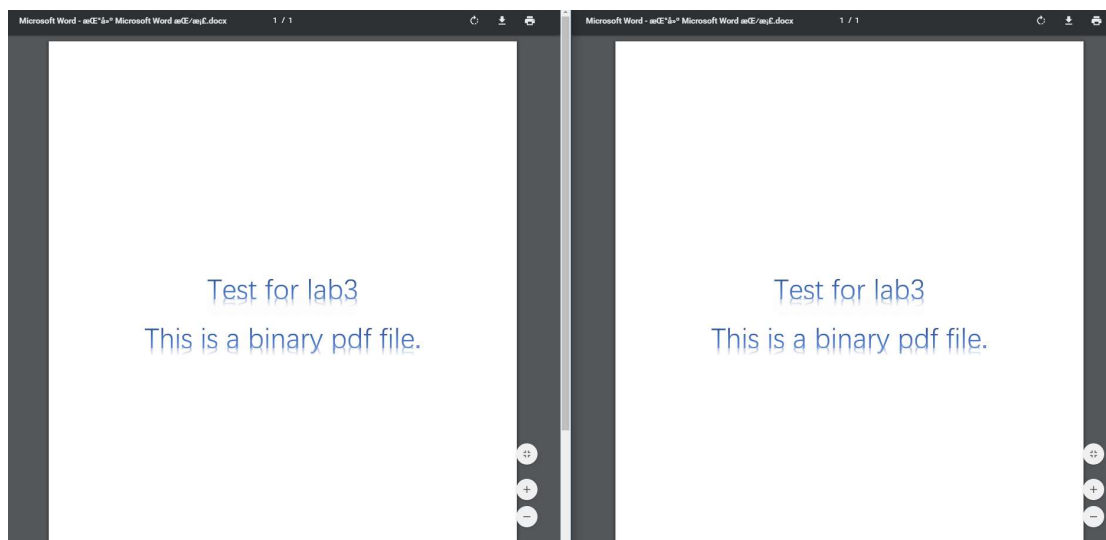


图 3.2 复制前后的二进制 PDF 文件

4 实验结果分析

1. 编译源程序:

```

Gray@MICRO50-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab3$ gcc lab3.c -o lab3.out -Wall -g -lpthread

```

图 3.3 gcc 编译截图

由截图可见，成功完成了编译。

2. 运行结果分析:

运行的过程如图 3.1 所示，程序提示誊抄成功。通过调用 ls 函数可以看到在目录中创建了指定的待写目标文件。如图 3.2 所示，两个实例 PDF 文件被成功地复制了。

5 心得体会与总结

1. 本次实验中比较有意思的环节是如何让从缓冲中复制数据到代写文件的进程知道什么时候应该停止下来，也就是两个进程通信的地方。在第一次写的时候，我采用了向缓冲区写入一个'\0'字符来作为标识符。在老师指出后，我意识到这样的方法并不适合二进制文件的复制。在第二次实现的

时候，我用了一个比较暴力的方法，即每个缓冲区的位置都存入一个完整的结构体，这个结构体带有数据和标志位，每次读取这个结构体的时候，都查看标志位以此来检查这个结构体是不是最后一个数据。这个方法的代价就是要花一半的共享内存空间在保存大量的标志位上。最后使用的方法则是上文中提到的只设置一个标志位的方式，用这个标志位来保存最后一个数据的索引，初始化的时候设置为-1，防止冲突。每次进程都去检查一下标志索引是不是和当前索引相同，若相同则说明读到了最后一个数据。

2. 缓冲技术的应用非常广泛，不仅在本次实验中的誊抄程序中，网络通信等等场景中都有大量的缓冲，能够有效地平衡不同设备的不同的速度，从而提高总效率。
3. Linux 中进程通信手段不像线程可以直接共享全局变量，需求还是比较苛刻。

实验四 TinyOS 实验

1 实验目的与要求

1. 了解典型 nesC 的程序结构及语法;
2. 了解 tinyos 执行机制, 实现程序异步处理的方法;
3. 了解 tinyos 中 task 抽象及其使用;
4. 在 Blink 程序中使用 printf 输出信息, 使用 task 实现计算和外部设备操作的并发;
5. 了解 Telosb 节点中传感器的类型与使用;
6. 了解 Telosb 节点的传感器数据的获取;
7. 获取的数据通过 printf 传输至电脑;
8. 将节点的传感器数据传输到基站, 并在电脑端解析显示, 了解数据的采集过程。

2 实验内容

1. (1) Blink 程序的编译和下载
(2) 给 Blink 程序加入 printf, 在每次定时器事件触发点亮 LED 的同时通过串口显示信息
(3) 修改 BLink 程序, 只使用一个 Timer, 三个 LED 灯作为 3 位的二进制数表示 (亮灯为 1, 不亮为 0), 按照 0-7 的顺序循环显示, 同时将数值显示在终端上。
2. 修改 Blink 程序, 在 timer0 的触发事件处理中加入计算。

```
event void Timer0.fired()
{
    uint32_t i;
    dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    for(i=0;i<400001;i++)
        call Leds.led0Toggle();
}
```

观察 LED 亮灯的情况, 分析其原因, 将 400001 改为 10001, 再观察并进

行分析，加入 `printf` 进行输出。

3. 采用 task 实现计算

```
uint32_t i;

task void computeTask()
{
    for (i=0; i 400001; i++) {}
}

event void Timer0.fired()
{
    dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    call Leds.led0Toggle();
    post computeTask();
}
```

观察 LED 亮灯的情况，分析其原因，将 400001 改为 10001，再观察并进行分析。

4. 请修改 `computetask` 的内容，将 400001 次计算分割成为若干小的部分，从而使得 LED1 和 LED2 的 `fire` 事件可以被正常调用，并通过 `printf` 输出。
5. 使用节点中的各个传感器，通过温度、湿度、光强强度计算，将数据反馈到电脑端进行输出。

3 实验过程与结果

1. 计数：
 - a) 基于原有的 `Blink` 例程，修改 `Makefile` 文件，添加加入添加 `printf` 组件必需的一些设置（包括缓冲区的大小，组件代码实现地址等）。
 - b) 在组件实现文件中加入，添加加入头文件声明，组件声明。
 - c) 修改主代码文件中，修改成只有一个计时器，每次计时完毕调用对应 `fired` 回调函数的时候，对一个全局计数全局变量进行自加。通过对这个全局变量的多个判断语句，对特定的 `Led` 灯组件调用 `led*On`, `led*Off` 函数来控制灯组的亮灭。
 - d) 运行结果：

```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink1_Count$ make install telosb /dev/ttyUSB0
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
gcc -o build/telosb/main.exe -Os -DNEW_PRINTF_SEMANTICS -fnesc-separator=__ -Wall -Wshadow -Wnes
c-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x22 -I/home/
gray/tinyos-main/tos/lib/printf -DPRINTF_BUFFER_SIZE=8 -DIDENT_APPNAME="\BlinkAppC\" -DIDENT_USER
NAME="\gray\" -DIDENT_HOSTNAME="\gray-VirtualBox\" -DIDENT_USERHASH=0x4e4c96a3L -DIDENT_TIMESTAMP
=0x5af54c24L -DIDENT_UIDHASH=0xdb7f4ba5L BlinkAppC.nc -lm
BlinkC.nc: In function 'BlinkC_Timer0_fired':
BlinkC.nc:68:3: warning: format '%d' expects argument of type 'int', but argument 2 has type 'uin
t32_t' [-Wformat]
compiled BlinkAppC to build/telosb/main.exe
8644 bytes in ROM
342 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
found mote on /dev/ttyUSB0 (using bsl,auto)
installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
8756 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out

```

图 4.1 程序编译烧录截图

```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink1_Count$ java net.tinyos.tools.PrintfClient -comm se
rial@/dev/ttyUSB0:telosb
Thread[Thread-1,5,main]serial@/dev/ttyUSB0:115200: resynchronising
5
6
7
0
1
2
3
4
5
6
7
0

```

图 4.2 电脑端程序监听串口数据截图

2. 循环:

- 在计数程序上进行修改，去除原自加计数的逻辑，在 fired 回调函数中添加一个循环次数为 400001 的 for 循环，以此模拟复杂计算的耗时。
- 将循环次数修改为 10001 后再次进行测试运行。
- 运行结果:

```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink2_Loop$ make install telosb /dev/ttyUSB0
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -DNEW_PRINTF_SEMANTICS -fnesc-separator=__ -Wall -Wshadow -Wnes
c-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x22 -I/home/
gray/tinyos-main/tos/lib/printf -DPRINTF_BUFFER_SIZE=128 -DIDENT_APPNAME="\BlinkAppC\" -DIDENT_US
ERNAME="gray\" -DIDENT_HOSTNAME="gray-VirtualBox\" -DIDENT_USERHASH=0x4e4c96a3L -DIDENT_TIMESTA
MP=0x5af54c83L -DIDENT_UIDHASH=0xb9d2f4d9L BlinkAppC.nc -lm
compiled BlinkAppC to build/telosb/main.exe
6262 bytes in ROM
482 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
found mote on /dev/ttyUSB0 (using bsl,auto)
installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
6432 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out

```

图 4.3 程序编译烧录截图

```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink2_Loop$ java net.tinyos.tools.PrintfClient -comm ser
ial@/dev/ttyUSB0:telosb
Thread[Thread-1,5,main]serial@/dev/ttyUSB0:115200: resynchronising
D uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0

```

图 4.4 电脑端程序监听串口数据（循环 400001 次）截图

3. 任务:

- 在循环程序上进行修改，将原有的 400001 次 for 循环的逻辑抽分到外部，通过调用一个 task 来进行实现。
- 将循环次数修改为 10001。
- 运行结果:


```

gray@gray-VirtualBox:~/tinys-main/apps/Blink3_Task$ make install telosb /dev/ttyUSB0
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -DNEW_PRINTF_SEMANTICS -fnesc-separator=__ -Wall -Wshadow -Wnes
c-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x22 -I/home/
gray/tinys-main/tos/lib/printf -DPRINTF_BUFFER_SIZE=128 -DIDENT_APPNAME="\BlinkAppC\" -DIDENT_US
ERNAME="gray\" -DIDENT_HOSTNAME="gray-VirtualBox\" -DIDENT_USERHASH=0x4e4c96a3L -DIDENT_TIMESTA
MP=0x5af54d07L -DIDENT_UIDHASH=0x81fec934L BlinkAppC.nc -lm
compiled BlinkAppC to build/telosb/main.exe
6270 bytes in ROM
482 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
found mote on /dev/ttyUSB0 (using bsl,auto)
installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
6442 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out

```

图 4.5 程序编译烧录截图

```

gray@gray-VirtualBox:~/tinys-main/apps/Blink3_Task$ java net.tinys.tools.PrintfClient -comm ser
ial@/dev/ttyUSB0:telosb
Thread[Thread-1,5,main]serial@/dev/ttyUSB0:115200: resynchronising
D uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED uint8: 0
here here is a LED

```

图 4.6 电脑端程序监听串口数据（循环 400001 次）截图

4. 切分任务：

- a) 在任务程序上进行修改，将 task 中的 400001 次循环模拟复杂计算的过程修改切分为 400 个循环 1000 次的小任务。从而使得程序的运行逻辑恢复正常。
- b) 运行结果：


```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink4_Split$ make install telosb /dev/ttyUSB0
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -DNEW_PRINTF_SEMANTICS -fnesc-separator=__ -Wall -Wshadow -Wnes
c-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x22 -I/home/
gray/tinyos-main/tos/lib/printf -DPRINTF_BUFFER_SIZE=128 -DIDENT_APPNAME="BlinkAppC" -DIDENT_US
ERNAME="gray\" -DIDENT_HOSTNAME="gray-VirtualBox\" -DIDENT_USERHASH=0x4e4c96a3L -DIDENT_TIMESTA
MP=0x5af54d3fL -DIDENT_UIDHASH=0x3b76930cL BlinkAppC.nc -lm
compiled BlinkAppC to build/telosb/main.exe
6294 bytes in ROM
484 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
found mote on /dev/ttyUSB0 (using bsl,auto)
installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
6466 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out

```

图 4.7 程序编译烧录截图

```

gray@gray-VirtualBox:~/tinyos-main/apps/Blink4_Split$ java net.tinyos.tools.PrintfClient -comm se
rial@/dev/ttyUSB0:telosb
Thread[Thread-1,5,main]serial@/dev/ttyUSB0:115200: resynchronising
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 2
here is a LED uint8: 0
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 0
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 2
here is a LED uint8: 0
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 0
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 2
here is a LED uint8: 0
here is a LED uint8: 0
here is a LED uint8: 1
here is a LED uint8: 0

```

图 4.8 电脑端程序监听串口数据截图

5. 传感器读取:

1. Printf 版本:

- AppC 配置文件中, 添加 DemoSensorC, SerialStartC, SensirionSht11C, HamamatsuS1087ParC, Printf 组件, 将程序与传感器的温度、湿度、光照器件链接。
- Makefile 中, 对 Printf 组件进行缓冲区大小、库文件路径等进行配置。
- 主程序文件中, booted 回调函数中为计时器设定频率, fired 回调函数中调用三个传感器部件的 read 功能。每个传感器的 readDone 回调中, 都在内部完成对读取值的预处理后, 通过调用 Printf, 通过串口向外输出字符串, 最后再修改 Led 灯的点亮情况。

2. ActiveMessage 版本:

- a) 在 Makefile 中, 为 JNI 通信配置配对指定的 nesC 结构体和 Java 类。
- b) 在 AppC 配置文件中, 添加 DemoSensorC, SensirionSht11C, HamamatsuS1087ParC, SerialActiveMessageC, SerialStartC 组件, 将程序与所需的组件们进行连接。对 ActiveMessage 进行管理、包等的配置。
- c) 主程序文件中, booted 回调中, 开启控制器。在控制器成功开启后为计时器初始化计时频率。
- d) 每次计时器 fired 回调后, 调用三个传感器的 read 功能。
- e) 每个传感器的 readDone 内, 调用 Packet.getPayload 获得一个待发送的包, 为包赋值为读取到的信息, 然后调用 AMSend.send 发送这个数据包, 切换对应的 Led 灯状态。
- f) 通过一个全局 flag 变量保证同一时间只有一个包被发送, 当包发送结束 (AMSend.sendDone 回调) 后才恢复全局 flag 变量。
- g) 电脑端监听的 Java Client 中, 通过命令行命令读取各项参数, 初始化包括监听在内的串口各项配置。在 messageReceived 回调函数中, 获得 TinyOS 发送的数据包, 根据不同的种类, 对数据值进行不同的处理, 最后完成输出。

```

/home/gray/tinyos-main/tos/chips/msp430/adc12/AdcStreamP.nc:191:20: warning: variable 'tmp_count'
set but not used [-Wunused-but-set-variable]
SensorC.nc: In function 'SensorC_readPhoto_readDone':
SensorC.nc:70:7: warning: format '%d' expects argument of type 'int', but argument 2 has type 'uint32_t' [-Wformat]
SensorC.nc:70:7: warning: format '%d' expects argument of type 'int', but argument 3 has type 'uint32_t' [-Wformat]
    compiled SensorAppC to build/telosb/main.exe
      18832 bytes in ROM
      622 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
  found mote on /dev/ttyUSB0 (using bsl,auto)
  installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
19440 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out

```

图 4.9 程序编译烧录截图

```

gray@gray-VirtualBox:~/HUST-OS-Experiments/lab_tinyOS/SensorDemo_printf$ java net.tinyos.tools.Pr
intfClient -comm serial@/dev/ttyUSB0:telosb
Thread[Thread-1,5,main]serial@/dev/ttyUSB0:115200: resynchronising
temp: 25.0
humidity: 69.0%
photo: 34.0 lux
temp: 25.0
humidity: 69.0%
photo: 34.0 lux
temp: 25.0
humidity: 69.0%
photo: 25.0 lux
temp: 25.0
humidity: 69.0%
photo: 34.0 lux
temp: 25.0
humidity: 69.0%
photo: 36.0 lux
temp: 25.0
humidity: 69.0%
photo: 27.0 lux
temp: 25.0
humidity: 69.0%
photo: 6.0 lux
temp: 25.0

```

图 4.10 电脑端程序监听串口数据截图

4 实验结果分析

1. 计数:

根据图 4.1,4.2 所示,程序成功编译烧录,从电脑端监听的串口数据来看,也成功读取到了 0-7 的循环计数。从传感器节点的 Led 灯组的结果来看,也成功按顺序展示了 0-7 对应的二进制码

2. 循环:

根据图 4.3,4.4 所示,程序成功编译烧录。从电脑端监听的串口数据来看,计时器每次只有 0 号被调用。从传感器节点的 Led 灯组的结果来看,也只有 0 号 Led 灯亮起。可见程序整个的时间安排,被复杂计算的时间完全打乱。

3. 任务:

根据图 4.5,4.6 所示,程序成功编译烧录。从电脑端监听的串口数据来看,计时器每次也只有 0 号被调用。从传感器节点的 Led 灯组的结果来看,也只有 0 号 Led 灯亮起。可见就算把复杂计算放在任务中,时间也被完全打乱。

4. 切分任务:

根据图 4.7,4.8 所示,程序成功编译烧录。从电脑端监听的串口数据来看,正常输出了计时器的轮转输出数据。从传感器节点啊的 Led 灯组来看,灯组亮起的频率也和程序中指定的频率相同。可见把复杂任务切分成多个小任务,解决了打乱时间线的问题。

5. 传感器读取:

根据图 4.9,4.10 所示,程序成功编译烧录,电脑监听的串口数据来看,也成功输出了正确的三个传感器读取的环境数据。从 Led 灯组的亮灭强

开来看，也遵循了正确的频率设置。用物体遮挡传感器后，感光数据也发生了从 40 到 6 lux 的变化（见图 4.10 的红圈位置）。

5 心得体会与总结

1. TinyOS 实验使用了 nesC 来进行开发，和 C 语言完全不同的设计方式，使用了大量的组件和回调。在前四个实验中，大部分的工作都是在原有的 Blink 上进行修改，其实只要理解程序的运行逻辑的前提下，修改起来其实并没有什么难度。
2. 读取传感器数据的实验中，直接采用 Printf 的方法比较简单，但是把计算预处理的过程放在了计算能力薄弱的 TinyOS 传感器节点中，而且 Printf 的格式化输出能力相较 C stdio 标准库的 printf 来说，缺乏很多功能，需求一些额外的代码来进行维护（比如支持输出小数）。
3. 在使用了 ActiveMessage 的版本中，通过了 JNI 生成包来进行通讯，在这个方案中，需要显式声明依赖大量的组件，写起来其实比较麻烦，但是把数据的计算处理过程放在了电脑端，并能传输更复杂的数据结构。
4. 其实 Printf 这个组件的功能已经提供了电脑端设备和 TinyOS 设备通过串口进行字符通信的功能，完全可以仿照网络通信中的方式进行一定简化后来进行通讯，比如把输出的信息格式化为 json 格式，在电脑 Java 端也能快速解译。

附加实验 Linux 文件目录

1 实验目的与要求

1. 了解 Linux 文件系统与目录操作；
2. 了解 Linux 文件系统目录结构；
3. 掌握文件和目录的程序设计方法。

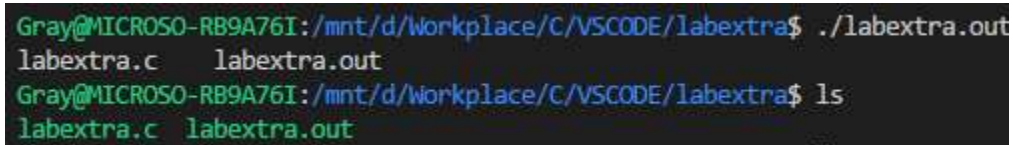
2 实验内容

编程实现目录查询功能：

- a) 功能类似 `ls -lR`；
- b) 查询指定目录下的文件及子目录信息，显示文件的类型、大小、时间等信息；
- c) 父递归显示子目录中的所有文件信息。

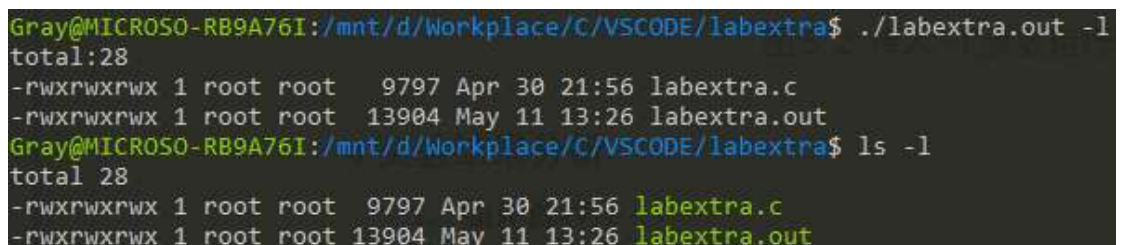
3 实验过程与结果

1. 从控制行命令中读取参数，以运行指定的逻辑（以下逻辑以传入了 `-lR` 为例）。
2. 调用 `opendir` 函数，传入指定路径的文件夹，来获得一个 `DIR` 指针，将这个指针循环传入 `readdir` 函数即可获得该文件夹内所有的文件路径，对循环体中遍历到的所有文件调用 `lstat` 获得这个文件的属性信息结构体，再调用 `getpwuid` 函数、`getgrgid` 函数、`localtime` 函数来获得更多数据，将所需数据整理为字符串存入缓存，若遍历到的文件为文件夹，则将这个文件夹路径存入缓存。
3. 通过缓存中的文件属性信息的数据计算占用硬盘块数，然后格式化数据输出。最后关闭当前查询的文件夹。
4. 对缓存中的所有未操作的文件夹文件，递归调用处理的过程（即）。
5. 运行结果：



```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ./labextra.out
labextra.c  labextra.out
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ls
labextra.c  labextra.out
```

图 5.1 无传入参数运行截图



```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ./labextra.out -l
total:28
-rwxrwxrwx 1 root root  9797 Apr 30 21:56 labextra.c
-rwxrwxrwx 1 root root 13904 May 11 13:26 labextra.out
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ls -l
total 28
-rwxrwxrwx 1 root root  9797 Apr 30 21:56 labextra.c
-rwxrwxrwx 1 root root 13904 May 11 13:26 labextra.out
```

图 5.2 传入'-l'参数运行截图

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ./labextra.out /mnt/d/Workplace/C/VSCODE
ODE
gcc.bat      lab1      lab2      lab3      labextra      lab_tinyOS  LICENSE
README.md
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ls /mnt/d/Workplace/C/VSCODE
gcc.bat  lab1  lab2  lab3  labextra  lab_tinyOS  LICENSE  README.md
```

图 5.3 传入指定路径参数运行截图

```
Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ./labextra.out /mnt/d/Workplace/C/VSCODE
ODE -lR
/mnt/d/Workplace/C/VSCODE:
total:12
-rwxrwxrwx 1 root root    77 Apr 13 15:18 gcc.bat
drwxrwxrwx 0 root root   512 May 10 00:22 lab1
drwxrwxrwx 0 root root   512 May 10 14:10 lab2
drwxrwxrwx 0 root root   512 May 11 13:06 lab3
drwxrwxrwx 0 root root   512 May 11 13:26 labextra
drwxrwxrwx 0 root root   512 May 08 20:04 lab_tinyOS
-rwxrwxrwx 1 root root  1060 Apr 10 23:57 LICENSE
-rwxrwxrwx 1 root root  1064 May 08 20:10 README.md

/mnt/d/Workplace/C/VSCODE/lab1:
total:16
-rwxrwxrwx 1 root root  1499 May 10 00:22 lab1.c
-rwxrwxrwx 1 root root 11016 May 10 00:22 lab1.out

/mnt/d/Workplace/C/VSCODE/lab2:
total:24
-rwxrwxrwx 1 root root  3437 May 10 00:43 lab2.c
-rwxrwxrwx 1 root root 16544 May 10 14:10 lab2.out

/mnt/d/Workplace/C/VSCODE/lab3:
total:108
-rwxrwxrwx 1 root root 81208 Apr 29 00:29 1.pdf
-rwxrwxrwx 1 root root  4335 May 10 17:04 lab3.c
-rwxrwxrwx 1 root root 17960 May 10 16:19 lab3.out

/mnt/d/Workplace/C/VSCODE/labextra:
total:28
-rwxrwxrwx 1 root root  9797 Apr 30 21:56 labextra.c
-rwxrwxrwx 1 root root 13904 May 11 13:26 labextra.out

/mnt/d/Workplace/C/VSCODE/lab_tinyOS:
total:0
drwxrwxrwx 0 root root   512 May 08 20:04 Blink1_Count
drwxrwxrwx 0 root root   512 May 08 20:04 Blink2_Loop
```

```

Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/labextra$ ls -lR /mnt/d/Workplace/C/VSCODE -lR
/mnt/d/Workplace/C/VSCODE:
total 8
-rwxrwxrwx 1 root root 77 Apr 13 15:18 gcc.bat
drwxrwxrwx 0 root root 512 May 10 00:22 lab1
drwxrwxrwx 0 root root 512 May 10 14:10 lab2
drwxrwxrwx 0 root root 512 May 11 13:06 lab3
drwxrwxrwx 0 root root 512 May 11 13:26 labextra
drwxrwxrwx 0 root root 512 May 8 20:04 lab_tinyOS
-rwxrwxrwx 1 root root 1060 Apr 10 23:57 LICENSE
-rwxrwxrwx 1 root root 1064 May 8 20:10 README.md

/mnt/d/Workplace/C/VSCODE/lab1:
total 16
-rwxrwxrwx 1 root root 1499 May 10 00:22 lab1.c
-rwxrwxrwx 1 root root 11016 May 10 00:22 lab1.out

/mnt/d/Workplace/C/VSCODE/lab2:
total 24
-rwxrwxrwx 1 root root 3437 May 10 00:43 lab2.c
-rwxrwxrwx 1 root root 16544 May 10 14:10 lab2.out

/mnt/d/Workplace/C/VSCODE/lab3:
total 108
-rwxrwxrwx 1 root root 81208 Apr 29 00:28 1.pdf
-rwxrwxrwx 1 root root 4335 May 10 17:04 lab3.c
-rwxrwxrwx 1 root root 17960 May 10 16:19 lab3.out

/mnt/d/Workplace/C/VSCODE/labextra:
total 28
-rwxrwxrwx 1 root root 9797 Apr 30 21:56 labextra.c
-rwxrwxrwx 1 root root 13904 May 11 13:26 labextra.out

/mnt/d/Workplace/C/VSCODE/lab_tinyOS:
total 0
drwxrwxrwx 0 root root 512 May 8 20:04 Blink1_Count
drwxrwxrwx 0 root root 512 May 8 20:04 Blink2_Loop
drwxrwxrwx 0 root root 512 May 8 19:52 Blink3_Task

```

图 5.4 传入'-lR'参数运行截图

4 实验结果分析

1. 编译源程序:

```

Gray@MICROSO-RB9A76I:/mnt/d/Workplace/C/VSCODE/lab1$ gcc lab1.c -o lab1.out

```

图 5.5 gcc 编译截图

由图 5.5 可见，使用 gcc 成功完成了编译过程。

2. 运行过程:

- 由图 5.1 可见，在无传入参数的状态下，输出了本文件夹下的所有文件名。
- 由图 5.2 可见，在传入'-l'的情况下，程序输出了本文件夹下的所有文件的详细信息，和 ls 程序的运行情况相同。
- 由图 5.3 可见，在传入指定文件夹路径的情况下，程序输出了该文件夹下的所有文件的文件名，且和 ls 程序的运行情况相同。
- 由图 5.4 可见，在传入'-lR'的情况下，程序递归输出了指定文件夹下的所有文件与文件夹的详细信息。

3. 压力测试:


```

-rw-rw-r-- 1 root root 1307 Mar 04 16:19 AccelConfigP.nc
-rw-rw-r-- 1 root root 1895 Mar 04 16:19 AccelP.nc
-rw-rw-r-- 1 root root 897 Mar 04 16:19 AccelReadP.nc
-rw-rw-r-- 1 root root 780 Mar 04 16:19 AccelReadStreamP.nc
-rw-rw-r-- 1 root root 831 Mar 04 16:19 AccelXC.nc
-rw-rw-r-- 1 root root 907 Mar 04 16:19 AccelXStreamC.nc
-rw-rw-r-- 1 root root 829 Mar 04 16:19 AccelYC.nc
-rw-rw-r-- 1 root root 907 Mar 04 16:19 AccelYStreamC.nc
-rw-rw-r-- 1 root root 336 Mar 04 16:19 ArbitratedPhotoDeviceP.nc
-rw-rw-r-- 1 root root 333 Mar 04 16:19 ArbitratedTempDeviceP.nc
-rw-rw-r-- 1 root root 634 Mar 04 16:19 DemoSensorC.nc
-rw-rw-r-- 1 root root 676 Mar 04 16:19 DemoSensorStreamC.nc
-rw-rw-r-- 1 root root 3201 Mar 04 16:19 Mag.nc
-rw-rw-r-- 1 root root 1440 Mar 04 16:19 MagConfigP.nc
-rw-rw-r-- 1 root root 2647 Mar 04 16:19 MagP.nc
-rw-rw-r-- 1 root root 2648 Mar 04 16:19 MagReadP.nc
-rw-rw-r-- 1 root root 2855 Mar 04 16:19 MagReadStreamP.nc
-rw-rw-r-- 1 root root 813 Mar 04 16:19 MagXC.nc
-rw-rw-r-- 1 root root 901 Mar 04 16:19 MagXStreamC.nc
-rw-rw-r-- 1 root root 813 Mar 04 16:19 MagYC.nc
-rw-rw-r-- 1 root root 897 Mar 04 16:19 MagYStreamC.nc
-rw-rw-r-- 1 root root 2121 Mar 04 16:19 MicC.nc
-rw-rw-r-- 1 root root 2598 Mar 04 16:19 MicDeviceP.nc
-rw-rw-r-- 1 root root 5224 Mar 04 16:19 MicP.nc
-rw-rw-r-- 1 root root 355 Mar 04 16:19 MicReadP.nc
-rw-rw-r-- 1 root root 533 Mar 04 16:19 MicReadStreamP.nc
-rw-rw-r-- 1 root root 4320 Mar 04 16:19 MicSetting.nc
-rw-rw-r-- 1 root root 2191 Mar 04 16:19 MicStreamC.nc
-rw-rw-r-- 1 root root 66 Mar 04 16:19 Mts300Sonder.nc
-rw-rw-r-- 1 root root 696 Mar 04 16:19 PhotoC.nc
-rw-rw-r-- 1 root root 1276 Mar 04 16:19 PhotoTempControlP.nc
-rw-rw-r-- 1 root root 1914 Mar 04 16:19 PhotoTempDeviceC.nc
-rw-rw-r-- 1 root root 928 Mar 04 16:19 PhotoTempP.nc
-rw-rw-r-- 1 root root 2792 Mar 04 16:19 SensorMts300C.nc
-rw-rw-r-- 1 root root 235 Mar 04 16:19 SounderC.nc
-rw-rw-r-- 1 root root 689 Mar 04 16:19 SounderP.nc
-rw-rw-r-- 1 root root 692 Mar 04 16:19 TempC.nc
-rw-rw-r-- 1 root root 2365 Mar 04 16:19 mts300.h

```

```

../../../../../../../../etc/apt/tinynos-main/tos/sensorboards/mts400:
total:152

```

```

-rw-rw-r-- 1 root root 1660 Mar 04 16:19 Accel202.h
-rw-rw-r-- 1 root root 2173 Mar 04 16:19 Accel202C.nc
-rw-rw-r-- 1 root root 2985 Mar 04 16:19 Accel202LogicP.nc
-rw-rw-r-- 1 root root 2798 Mar 04 16:19 Accel202ReaderP.nc
-rw-rw-r-- 1 root root 2217 Mar 04 16:19 Adg715.h
-rw-rw-r-- 1 root root 4953 Mar 04 16:19 Adg715C.nc
-rw-rw-r-- 1 root root 2905 Mar 04 16:19 Adg715ControlC.nc
-rw-rw-r-- 1 root root 7336 Mar 04 16:19 Adg715ControlP.nc
-rw-rw-r-- 1 root root 2214 Mar 04 16:19 Calibration.nc
-rw-rw-r-- 1 root root 2540 Mar 04 16:19 Channel.nc
-rw-rw-r-- 1 root root 2176 Mar 04 16:19 HalAccel202C.nc
-rw-rw-r-- 1 root root 2286 Mar 04 16:19 HalIntersema5534C.nc
-rw-rw-r-- 1 root root 2662 Mar 04 16:19 HalSensirionSht11C.nc
-rw-rw-r-- 1 root root 2065 Mar 04 16:19 HalTaos2550C.nc
-rw-rw-r-- 1 root root 2388 Mar 04 16:19 HplAccel202C.nc
-rw-rw-r-- 1 root root 4052 Mar 04 16:19 HplAccel202P.nc
-rw-rw-r-- 1 root root 2763 Mar 04 16:19 HplAdg715C.nc
-rw-rw-r-- 1 root root 2729 Mar 04 16:19 HplIntersema5534C.nc
-rw-rw-r-- 1 root root 4519 Mar 04 16:19 HplIntersema5534P.nc

```


图 5.6 传入根目录作为指定路径的参数

由图 5.6 可见，程序成功递归输出了根目录下的所有的文件的详细信息。

5 心得体会与总结

1. ls 在传入'-l'参数的时候，会在输出文件夹下文件的详细信息前，会输出本文件夹内所有文件所占用的硬盘块数。在模拟 ls 程序的过程中，我在计算总块数的时候遇到了不小的问题，通过逐渐的摸索才总结出 ls 命令在管理文件的时候，默认把硬盘的单块体积设为 4096 字节，同时计算时默认忽略文件夹、链接文件的体积，过小体积的文件也不被计入。
2. 文件系统是 Linux 系统中比较复杂的一个环节，在 C 语言中的库函数系统调用中，整个调用的思路也设计得比较复杂。在搜索了一些有关的例程后，才大概清楚了文件描述结构体的作用。

附录

lab1.c:

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int child1(int *filedis);
int child2(int *filedis);
int handle();
int pid1 = 0;
int pid2 = 0;
int pipe_field[2];// 一读一写 0 写 1 读

int main()
{
    // signal(SIGINT,SIG_IGN);
    if (pipe(pipe_field) < 0)
    {
        printf("pipe create failed!\n");
        return -1;
    }
    else
    {
        printf("-----successfully-----\n");

        pid1 = fork();
        if (pid1 == 0)
        {
            child1(pipe_field);
            // exit(0);
        }
        pid2 = fork();
        if (pid2 == 0)
        {
            child2(pipe_field);
            // exit(0);
        }

        //parent
```

```

        signal(SIGINT,handle);
        wait(pid1);
        wait(pid2);

        //close pipe
        close(pipe_field[0]);
        close(pipe_field[1]);

        printf("Parent Process is Killed!\n");
    }
    return 0;
}

int handle(){
    kill(pid1, SIGKILL);
    printf("Child Process 1 is Killed by Parent!\n");
    kill(pid2, SIGKILL);
    printf("Child Process 2 is Killed by Parent!\n");
}

int child1(int *filedis)
{
    int count = 0;
    close(filedis[0]);
    while (1)
    {
        // printf("1 is in loops\n");
        char string[50];
        sprintf(string, "I send you %d", count);
        strcat(string, " times.\n");
        write(filedis[1], string, 50);
        sleep(1);
        count++;
    }
    close(filedis[1]);
}

int child2(int *filedis)
{
    char cache[50];
    cache[0] = 0;
    close(filedis[1]);
    while (1)
    {

```

```

        // printf("1 is in loops\n");
        read(filedis[0], cache, 50);
        printf("%s",cache);
    }
    close(filedis[0]);
}

```

lab2.c:

```

#include <fcntl.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/sem.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

void printTid();
void *myThread();
void delSemvalue();
int setSemvalue();
void *subp1();
void *subp2();
int P(int index);
int V(int index);

pthread_t id1; //thread id
pthread_t id2;
int sem_id = 0;
int threadCount = 2;

union semun {
    int val;
    struct semid_ds *buf;
    unsigned short *array;
};

// struct sembuf
// {
//     short sem_num; // index
//     short sem_op;  // Operation for semaphore, -1->P, +1->V.
//     short sem_flg; // Operation flags. Let OS know where is this sem, and
//                     delete it when something bad happens

```

```

// };

int main()
{
    printf("-----hello world!-----\n");

    //inititalize semaphore (we have two)
    sem_id = semget((key_t)IPC_PRIVATE, 2, 0666 | IPC_CREAT);
    if (sem_id == -1)
    {
        printf("failed to initalize semaphore\n");
        exit(0);
    }
    if (setSemvalue() == 0)
    {
        printf("setSemvalue failed\n");
        exit(1);
    }
    printf("successfully initalized\n");

    int result1 = pthread_create(&id1, NULL, subp1, NULL);
    int result2 = pthread_create(&id2, NULL, subp2, NULL);
    if (result1 != 0 || result2 != 0)
    {
        printf("result failed\n");
        exit(1);
    }
    printf("threads have been created\n");
    // printTid();
    // while (threadCount > 0)
    // {
    //     // sleep(2);
    // }
    void *status[2];
    pthread_join(id1, &status[0]);
    pthread_join(id2, &status[1]);

    delSemvalue(); //delete
    return 0;
}

//init sem's value
int setSemvalue()
{

```

```

    union semun arg1;
    arg1.val = 0;
    union semun arg2;
    arg2.val = 1;
    if (semctl(sem_id, 0, SETVAL, arg1) == -1)
    {
        printf("setSemvalue failed\n");
        return 0;
    }
    if (semctl(sem_id, 1, SETVAL, arg2) == -1)
    {
        printf("setSemvalue failed\n");
        return 0;
    }
    return 1;
}

void delSemvalue()
{
    union semun sem_union;
    if (semctl(sem_id, 1, IPC_RMID, sem_union) == -1)
    {
        printf("delSemvalue failed\n");
        exit(1);
    }
}

int sum = 0;

void *subp1()
{
    printTid();
    int i;
    for (i = 1; i <= 100; i++)
    {
        P(1);
        // printf("thread 1 in loops, %d times\n", i);
        sum += i;
        V(0);
    }
    // threadCount--;
    return ((void *)0);
}

```

```

void *subp2()
{
    printTid();
    int i;
    for (i = 1; i <= 100; i++)
    {
        P(0);
        // printf("thread 2 in loops, %d times\n", i);
        printf("%d\n", sum);
        V(1);
    }
    // threadCount--;
    return ((void *)0);
}

void printTid()
{
    printf("tid: (0x%x)\n", (unsigned long)pthread_self());
}

int P(int index)
{
    //add 1 to sem
    struct sembuf sem_b;
    sem_b.sem_num = index;
    sem_b.sem_op = -1;
    sem_b.sem_flg = 0;
    if (semop(sem_id, &sem_b, 1) == -1)
    {
        printf("P failed\n");
        return 0;
    }
    return 1;
}

int V(int index)
{
    //delete 1 from sem
    struct sembuf sem_b;
    sem_b.sem_num = index;
    sem_b.sem_op = 1;
    sem_b.sem_flg = 0;
    if (semop(sem_id, &sem_b, 1) == -1)
    {

```

```

        printf("V failed\n");
        return 0;
    }
    return 1;
}

```

lab3.c

```

#include <fcntl.h>
#include <pthread.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
// #include <sys/wait.h>

#define LENGTH 128

int pid1;
int pid2;
int sem_id = 0;
int shmid;
int shmid2;
FILE* readFile;
FILE* writeFile;

void writeChild();
void readChild();
int setSemvalue();
void delSemvalue();
int P(int index);
int V(int index);

union semun {
    int val;
    struct semid_ds* buf;
    unsigned short* arry;
};

int main(int argc, char* argv[]) {

```



```

printf("-----start copying!-----\n");

if (argc != 3) {
    printf("args error\n");
    exit(1);
}

readFile = fopen(argv[1], "rb");
writeFile = fopen(argv[2], "wb");
if (readFile == NULL || writeFile == NULL) {
    printf("File error.\n");
    exit(1);
}

// initialize shared memory
if ((shmid = shmget(IPC_PRIVATE, (LENGTH + 1) * sizeof(char),
                    0666 | IPC_CREAT)) == -1) // connect
{
    printf("Create Share Memory Error");
    exit(1);
}

// initialize sem
sem_id = semget((key_t)IPC_PRIVATE, 2, 0666 | IPC_CREAT);
if (sem_id == -1) {
    printf("failed to initialize semaphore\n");
    exit(0);
}
if (setSemvalue() == 0) {
    exit(1);
}

printf("successfully initialized\n");

// initialize process
pid1 = fork();
if (pid1 == 0) {
    readChild();
    exit(0);
}
pid2 = fork();
if (pid2 == 0) {
    writeChild();
    exit(0);
}

```

```

    }

    wait(pid1);
    wait(pid2);

    printf("finish copying\n");
    delSemvalue();
    // shmdt(head_addr);//deattach
    if (shmctl(shmid, IPC_RMID, 0) < 0) // release shared memory
    {
        printf("release error\n");
        exit(1);
    }

    fclose(readFile);
    fclose(writeFile);
    return 0;
}

```

```

void writeChild() {
    char* head_addr = (char*)shmat(shmid, 0, 0);
    head_addr[LENGTH] = -1;
    int index = 0;
    // write
    char get = ' ';

    while (fread(&get, sizeof(char), 1, readFile) != 0) {
        P(1); // init with 1024
        head_addr[index] = get;
        // printf("write %d\n", index);
        index++;
        index = index % LENGTH;
        if (feof(readFile))
            head_addr[LENGTH] = index;
        V(0);
    }
    // P(0);
    // printf("write end\n");
    // head_addr[LENGTH] = index-1;
    // V(0);
}

```

```

void readChild() {
    char* head_addr = (char*)shmat(shmid, 0, 0);

```

```

int index = 0;
// read
while (1) {
    P(0); // init with 0
    char get = head_addr[index];
    fwrite(&get, sizeof(char), 1, writeFile);
    if (index == head_addr[LENGTH]) {
        // printf("read end\n");
        break;
    }
    index++;
    index = index % LENGTH;

    V(1);
}

// init sem's value
int setSemvalue() {
    union semun arg1;
    arg1.val = 0; // first one refers to "place to write"
    union semun arg2;
    arg2.val = LENGTH; // refer to "place to read"
    if (semctl(sem_id, 0, SETVAL, arg1) == -1) {
        printf("setSemvalue failed\n");
        return 0;
    }
    if (semctl(sem_id, 1, SETVAL, arg2) == -1) {
        printf("setSemvalue failed\n");
        return 0;
    }
    return 1;
}

void delSemvalue() {
    union semun sem_union;
    if (semctl(sem_id, 1, IPC_RMID, sem_union) == -1) {
        printf("delSemvalue failed\n");
        exit(1);
    }
}

int P(int index) {
    // add 1 to sem

```

```

    struct sembuf sem_b;
    sem_b.sem_num = index;
    sem_b.sem_op = -1;
    sem_b.sem_flg = 0;
    if (semop(sem_id, &sem_b, 1) == -1) {
        printf("P failed %d\n", index);
        return 0;
    }
    return 1;
}

int V(int index) {
    // delete 1 from sem
    struct sembuf sem_b;
    sem_b.sem_num = index;
    sem_b.sem_op = 1;
    sem_b.sem_flg = 0;
    if (semop(sem_id, &sem_b, 1) == -1) {
        printf("V failed %d\n", index);
        return 0;
    }
    return 1;
}

```

Blink1_Count_AppC.nc:

```

#include "printf.h"

configuration BlinkAppC
{
}
implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components PrintfC;
    components SerialStartC;

    BlinkC -> MainC.Boot;

    BlinkC.Timer0 -> Timer0;
    BlinkC.Leds -> LedsC;
}

```

Blink1_Count_C.nc

```
#include "Timer.h"

module BlinkC @safe()
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Leds;
    uses interface Boot;
}
implementation
{
    uint32_t i = 0;
    event void Boot.booted()
    {
        call Timer0.startPeriodic( 1000 );
    }

    event void Timer0.fired()
    {
        dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());

        printf("%d\n",i);

        if((i&1) == 1){ call Leds.led0On();}
        else call Leds.led0Off();

        if((i&2) == 2){ call Leds.led1On();}
        else call Leds.led1Off();

        if((i&4) == 4){ call Leds.led2On();}
        else call Leds.led2Off();

        i++;

        if(i == 8) i = 0;
    }
}
```

Blink2_Loop_AppC.nc:

```
#include "printf.h"
```

```

configuration BlinkAppC
{
}
implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;
    components PrintfC;
    components SerialStartC;

    BlinkC -> MainC.Boot;

    BlinkC.Timer0 -> Timer0;
    BlinkC.Timer1 -> Timer1;
    BlinkC.Timer2 -> Timer2;
    BlinkC.Leds -> LedsC;
}

```

Blink2_ Loop _C.nc

```

#include "Timer.h"

module BlinkC @safe()
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Timer<TMilli> as Timer2;
    uses interface Leds;
    uses interface Boot;
}
implementation
{
    uint32_t i = 0;
    event void Boot.booted()
    {
        call Timer0.startPeriodic( 250 );
        call Timer1.startPeriodic( 500 );
        call Timer2.startPeriodic( 1000 );
    }

    event void Timer0.fired()
    {
        dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    }
}

```

```

    for(i = 0;i<10001;i++){
        call Leds.led0Toggle();
        printf("here is a LED uint8: 0\n");
        printf fflush();
    }
}

event void Timer1.fired()
{
    dbg("BlinkC", "Timer 1 fired @ %s \n", sim_time_string());
    call Leds.led1Toggle();
    printf("here is a LED uint8: 1\n");
    printf fflush();
}

event void Timer2.fired()
{
    dbg("BlinkC", "Timer 2 fired @ %s.\n", sim_time_string());
    call Leds.led2Toggle();
    printf("here is a LED uint8: 2\n");
    printf fflush();
}
}

```

Blink3_Task_AppC.nc:

```

#include "printf.h"

configuration BlinkAppC
{
}

implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;
    components PrintfC;
    components SerialStartC;

    BlinkC -> MainC.Boot;

    BlinkC.Timer0 -> Timer0;

```

```

    BlinkC.Timer1 -> Timer1;
    BlinkC.Timer2 -> Timer2;
    BlinkC.Leds -> LedsC;
}

```

Blink3_Task_C.nc

```

#include "Timer.h"

```

```

module BlinkC @safe()

```

```

{
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Timer<TMilli> as Timer2;
    uses interface Leds;
    uses interface Boot;
}

```

```

implementation

```

```

{
    uint32_t i;
    task void computeTask(){
        for(i=0;i<400001;i++){//simulate compute cost
        }
}

```

```

event void Boot.booted()

```

```

{
    call Timer0.startPeriodic( 250 );
    call Timer1.startPeriodic( 500 );
    call Timer2.startPeriodic( 1000 );
}

```

```

event void Timer0.fired()

```

```

{
    dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    printf("here is a LED uint8: 0\n");
    printfflush();
    post computeTask();//without this line this program would be normal Blink
    call Leds.led0Toggle();
}

```

```

event void Timer1.fired()

```

```

{
    dbg("BlinkC", "Timer 1 fired @ %s \n", sim_time_string());
    call Leds.led1Toggle();
}

```



```

        printf("here is a LED uint8: 1\n");
        printf.flush();
    }

    event void Timer2.fired()
    {
        dbg("BlinkC", "Timer 2 fired @ %s.\n", sim_time_string());
        call Leds.led2Toggle();
        printf("here is a LED uint8: 2\n");
        printf.flush();
    }
}

```

Blink4_Split_AppC.nc:

```

#include "printf.h"

configuration BlinkAppC
{
}
implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;
    components PrintfC;
    components SerialStartC;

    BlinkC -> MainC.Boot;

    BlinkC.Timer0 -> Timer0;
    BlinkC.Timer1 -> Timer1;
    BlinkC.Timer2 -> Timer2;
    BlinkC.Leds -> LedsC;
}

```

Blink4_Split_C.nc:

```

#include "Timer.h"

module BlinkC @safe()
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
}

```

```

    uses interface Timer<TMilli> as Timer2;
    uses interface Leds;
    uses interface Boot;
}

implementation
{
    uint32_t start;

    task void computeSmallTask(){
        uint32_t temp = 0;
        //int variable start would be updated in another func
        for(temp = start;temp<start+1000;temp++){
            //do computing
        }
    }

    task void computeTask(){
        uint32_t max = 400001;
        uint32_t iteration = max/1000;
        uint32_t i = 0;

        for(i=0;i<iteration;i++){
            start = i*1000;
            post computeSmallTask();
        }

        start = (i+1)*100;

        for(i = iteration*1000;i<max;i++){
            //do computing
        }

    }

    event void Boot.booted()
    {
        call Timer0.startPeriodic( 250 );
        call Timer1.startPeriodic( 500 );
        call Timer2.startPeriodic( 1000 );
    }

    event void Timer0.fired()
    {

```

```

    dbg("BlinkC", "Timer 0 fired @ %s.\n", sim_time_string());
    printf("here is a LED uint8: 0\n");
    printf fflush();
    post computeTask();//without this line this program would be normal Blink
    call Leds.led0Toggle();
}

event void Timer1.fired()
{
    dbg("BlinkC", "Timer 1 fired @ %s \n", sim_time_string());
    call Leds.led1Toggle();
    printf("here is a LED uint8: 1\n");
    printf fflush();
}

event void Timer2.fired()
{
    dbg("BlinkC", "Timer 2 fired @ %s.\n", sim_time_string());
    call Leds.led2Toggle();
    printf("here is a LED uint8: 2\n");
    printf fflush();
}
}

```

Makefile(Blink1-4):

```

CFLAGS += -I$(TOSDIR)/lib/printf
PFLAGS += -DNEW_PRINTF_SEMANTICS
CFLAGS += -DPRINTF_BUFFER_SIZE=128

```

```

COMPONENT=BlinkAppC
include $(MAKERULES)

```

Sensor_Printf:

Makefile:

```

CFLAGS += -I$(TOSDIR)/lib/printf
PFLAGS += -DNEW_PRINTF_SEMANTICS
CFLAGS += -DPRINTF_BUFFER_SIZE=128
COMPONENT=SensorAppC

```

```

include $(MAKERULES)

```

SensorAppC.nc:

```

#include "printf.h"

```

```

configuration SensorAppC
{
}

implementation
{
    components MainC, SensorC, LedsC;
    components new TimerMilliC() as Timer0;
    components PrintfC;
    components SerialStartC;

    components new DemoSensorC() as Sensor;

    components new SensirionSht11C();
    components new HamamatsuS1087ParC();

    SensorC -> MainC.Boot;

    SensorC.readTemp -> SensirionSht11C.Temperature;
    SensorC.readHumidity -> SensirionSht11C.Humidity;
    SensorC.readPhoto -> HamamatsuS1087ParC;

    SensorC.Timer0 -> Timer0;
    SensorC.Leds -> LedsC;
}

```

SensorC.nc

```

#include "Timer.h"
#include "SensirionSht11.h"

/*read and print raw data without any process*/

module SensorC @safe()
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Leds;
    uses interface Boot;
    uses interface Read<uint16_t> as readTemp;
    uses interface Read<uint16_t> as readHumidity;
    uses interface Read<uint16_t> as readPhoto;
}

implementation

```

```

{
    #define FREQUENCY 1000

    uint16_t tempData;
    uint16_t humidityData;
    uint16_t photoData;
    bool locked =FALSE;

    event void Boot.booted()
    {
        call Timer0.startPeriodic(FREQUENCY);
    }

    event void Timer0.fired()
    {
        call readTemp.read();
        call readHumidity.read();
        call readPhoto.read();
    }

    event void readTemp.readDone(error_t result, uint16_t val){
        if(result == SUCCESS){
            double temp = 10*(-40.1+0.01*val);
            uint32_t a = temp/10;
            uint32_t b = temp-temp/10*10;
            printf("temp: %d.%d\n",a,b);
            printf fflush();
            call Leds.led0Toggle();
        }
    }

    event void readHumidity.readDone(error_t result, uint16_t val){
        if(result == SUCCESS){
            double temp = (-4+0.0405*val+(-2.8/1000000)*val*val)*10;
            uint32_t a = temp/10;
            uint32_t b = temp-temp/10*10;

            printf("humidity: %d.%d%\n",a,b);
            printf fflush();
            call Leds.led1Toggle();
        }
    }
}

```

```

event void readPhoto.readDone(error_t result, uint16_t val){
    if(result == SUCCESS){
        double temp = 0.0625*1000000*(val*1.5/4096/10000)*1000*10;
        uint32_t a = temp/10;
        uint32_t b = temp-temp/10*10;
        printf("photo: %d.%d lux\n",a,b);
        printf fflush();
        call Leds.led2Toggle();
    }
}
}

```

Sensor_AM:

Makefile:

```

CFLAGS += -I$(TOSDIR)/lib/printf
PFLAGS += -DNEW_PRINTF_SEMANTICS
CFLAGS += -DPRINTF_BUFFER_SIZE=128
COMPONENT=SensorAppC

```

```

BUILD_EXTRA_DEPS += SensorMsg.class
CLEAN_EXTRA = *.class SensorMsg.java

```

```

SensorMsg.class: SensorMsg.java
    javac SensorMsg.java

```

SensorMsg.java:

```

    mig java -target=$(PLATFORM) -java-classname=SensorMsg Sense.h
    SensorMsg -o $@

```

```

include $(MAKERULES)

```

SensorAppC.nc:

```

#include "printf.h"

```

```

configuration SensorAppC
{
}

```

```

implementation
{

```

```

components MainC, SensorC, LedsC;
components new TimerMilliC() as Timer0;
components PrintfC;
components SerialStartC;

components new DemoSensorC() as Sensor;

components new SensirionSht11C();
components new HamamatsuS1087ParC();

components SerialActiveMessageC as AM;

SensorC -> MainC.Boot;

SensorC.readTemp -> SensirionSht11C.Temperature;
SensorC.readHumidity -> SensirionSht11C.Humidity;
SensorC.readPhoto -> HamamatsuS1087ParC;

SensorC.Timer0 -> Timer0;
SensorC.Leds -> LedsC;

SensorC.Packet -> AM;
SensorC.AMPacket -> AM;
SensorC.Control -> AM;
SensorC.AMSend -> AM.AMSend[AM_SENSORMSG];
}

```

SensorC.nc:

```

#include "Timer.h"
#include "Sensor.h"
#include "SensirionSht11.h"

/*read and print raw data without any process*/

module SensorC @safe()
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Leds;
    uses interface Boot;
    uses interface Read<uint16_t> as readTemp;
    uses interface Read<uint16_t> as readHumidity;
    uses interface Read<uint16_t> as readPhoto;
}

```

```

    uses interface Packet;
    uses interface AMPacket;
    uses interface AMSend;
    uses interface SplitControl as Control;
}

```

implementation

```

{
    #define FREQUENCY 100

    uint16_t tempData;
    uint16_t humidityData;
    uint16_t photoData;

    message_t packet;
    bool locked =FALSE;

    event void Boot.booted()
    {
        // call Timer0.startPeriodic(FREQUENCY);
        call Control.start();
    }

    event void Control.startDone(error_t err)
    {
        if (err == SUCCESS)
            call Timer0.startPeriodic(1000);
    }

    event void Control.stopDone(error_t err)
    {
        //do sth
    }

    event void Timer0.fired()
    {
        if(locked){
            return;
        }
        call readTemp.read();
        call readHumidity.read();
        call readPhoto.read();
    }
}

```



```

event void readTemp.readDone(error_t result, uint16_t val){
    if(result == SUCCESS){
        SensorMsg *payload = (SensorMsg*) call Packet.getPayload(&packet,
sizeof(SensorMsg));
        if(!payload)return;

        payload->nodeid = TOS_NODE_ID;
        payload->kind = TEMPORARY;//0
        payload->data = val;

        if (call AMSend.send(AM_BROADCAST_ADDR, &packet,
sizeof(SensorMsg)) == SUCCESS)
        {
            call Leds.led0Toggle();
            locked = TRUE;
        }
    }
}

```

```

event void readHumidity.readDone(error_t result, uint16_t val){
    if(result == SUCCESS){
        SensorMsg *payload = (SensorMsg*) call Packet.getPayload(&packet,
sizeof(SensorMsg));
        if(!payload)return;

        payload->nodeid = TOS_NODE_ID;
        payload->kind = HUMIDITY;//1
        payload->data = val;

        if (call AMSend.send(AM_BROADCAST_ADDR, &packet,
sizeof(SensorMsg)) == SUCCESS)
        {
            call Leds.led1Toggle();
            locked = TRUE;
        }
    }
}

```

```

event void readPhoto.readDone(error_t result, uint16_t val){
    if(result == SUCCESS){
        SensorMsg *payload = (SensorMsg*) call Packet.getPayload(&packet,
sizeof(SensorMsg));
        if(!payload)return;

```

```

        payload->nodeid = TOS_NODE_ID;
        payload->kind = 2;//2
        payload->data = val;

        if (call AMSend.send(AM_BROADCAST_ADDR, &packet,
sizeof(SensorMsg)) == SUCCESS)
        {
            call Leds.led2Toggle();
            locked = TRUE;
        }
    }
}

event void AMSend.sendDone(message_t* msg, error_t err)
{
    if (&packet == msg)
    {
        locked = FALSE;
    }
}
}

```

Sensor.h:

```

#ifndef SENSOR_H
#define SENSOR_H

#define TEMPORARY 0
#define HUMIDITY 1
#define PHOTOVOLTAIC 2

typedef nx_struct SensorMsg{
    nx_uint16_t nodeid;
    nx_uint16_t kind;
    nx_uint16_t data;
} SensorMsg;

enum{
    AM_SENSORMSG = 6,
};

#endif

```

SensorClient.java:

```

import java.io.IOException;

import net.tinyos.message.*;
import net.tinyos.packet.*;
import net.tinyos.util.*;

public class SensorClient implements MessageListener {
    private MotelF motelF;

    public SensorClient(MotelF motelF) {
        this.motelF = motelF;
        this.motelF.registerListener(new SensorMsg(), this);
    }

    public static void main(String[] args) throws Exception {
        String source = null;
        if (args.length == 2) {
            if (!args[0].equals("-comm")) {
                usage();
                System.exit(1);
            }
            source = args[1];
        } else if (args.length != 0) {
            usage();
            System.exit(1);
        }

        PhoenixSource phoenix;

        if (source == null) {
            phoenix =
BuildSource.makePhoenix(PrintStreamMessenger.err);
        } else {
            phoenix = BuildSource.makePhoenix(source,
PrintStreamMessenger.err);
        }

        MotelF mif = new MotelF(phoenix);
        SensorClient serial = new SensorClient(mif);
    }

    public void messageReceived(int to, Message message) {
        SensorMsg msg = (SensorMsg) message;
        int type = msg.get_kind();
    }
}

```

```

        double temperature;
        double humidity;
        double photo;

        switch (type) {
        case 0:
            temperature = -40.1 + 0.01 * msg.get_data();
            System.out.println("Temperature:" + temperature + "°C");
            break;
        case 1:
            humidity = -4 + 0.0405 * msg.get_data() + (-2.8 / 1000000) *
msg.get_data() * msg.get_data();
            System.out.println("Humidity:" + humidity + "%");
            break;
        case 2:
            photo = msg.get_data() * 1.5 / 4096 / 10000;
            photo = 0.625 * 1000000 * photo * 1000;
            System.out.println("Photo:" + photo + "Lux");
            break;
        default:
            System.out.println("Unknow data:" + msg.get_data());
            break;
        }

        // try {
        //     Thread.sleep(1000);
        // } catch (Exception e) {}

    }

    private static void usage() {
        System.err.println("usage: SensorClient [-comm <source>]");
    }

}

```

labextra.c:

```

#include <dirent.h>
#include <grp.h>
#include <pwd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>

```

```

#include <sys/stat.h>
#include <sys/types.h>
#include <termios.h>
#include <time.h>
#include <unistd.h>

int showDir(char* dirname, int mode);
char* getFileInfo(struct stat* sP, char* filename, int* blocks);
char* num2month(int num);
void printInfoList(char** allFileInfos, int indexForInfos);
void printFormatList(char** allFileNames,
                    int maxLengthOfFileName,
                    int indexForFileNames);

int WIDTH = 0;  // num of columns of terminal

int main(int argc, char** argv) {
    char* dir = ".";
    int mode = 0;

    // get width(Columns) of terminal
    struct winsize size;
    ioctl(STDIN_FILENO, TIOCGWINSZ, &size);
    WIDTH = size.ws_col;

    // read arg's info
    if (argc != 1) {
        for (int i = 1; i < argc; i++) {
            if (argv[i][0] == '-') {
                if (strcmp("-lR", argv[i]) == 0) {
                    mode = 2;
                } else {
                    mode = 1;
                }
            } else {
                dir = argv[i];  // change the path
            }
        }
    }

    showDir(dir, mode);

    return 0;
}

```

```

// called by showDir
// with '-l' further information of files
char* getFileInfo(struct stat* sP, char* filename, int* blocks) {
    char* buf = (char*)malloc(sizeof(char) * 1024);
    struct stat s = *sP;

    switch (s.st_mode & S_IFMT) {
        case S_IFREG:
            sprintf(buf, "-");
            break;
        case S_IFDIR:
            sprintf(buf, "d");
            break;
        case S_IFLNK:
            sprintf(buf, "l");
            break;
        case S_IFBLK:
            sprintf(buf, "b");
            break;
        case S_IFCHR:
            sprintf(buf, "c");
            break;
        case S_IFIFO:
            sprintf(buf, "p");
            break;
        case S_IFSOCK:
            sprintf(buf, "s");
            break;
    }

    for (int i = 8; i >= 0; i--) {
        if (s.st_mode & (1 << i)) {
            switch (i % 3) {
                case 2:
                    strcat(buf, "r");
                    break;
                case 1:
                    strcat(buf, "w");
                    break;
                case 0:
                    strcat(buf, "x");
                    break;
            }
        }
    }
}

```

```

        } else {
            strcat(buf, "-");
        }
    }

    struct passwd* p = getpwuid(s.st_uid);
    struct group* g = getgrgid(s.st_gid);

    char temp[128];
    sprintf(temp, " %d %s %s %6ld", (int)s.st_nlink, p->pw_name, g->gr_name,
        s.st_size);
    strcat(buf, temp);

    struct tm* t = localtime(&s.st_ctime);
    sprintf(temp, " %s %02d %02d:%02d", num2month(t->tm_mon + 1),
t->tm_mday,
        t->tm_hour, t->tm_min);
    strcat(buf, temp);

    sprintf(temp, "%s\n", filename);
    strcat(buf, temp);

    int now4Blocks = s.st_size / 4096;
    if (S_ISLNK(s.st_mode) || S_ISDIR(s.st_mode)) {
        now4Blocks = 0;
    } else if (s.st_size % 4096 != 0) {
        now4Blocks++;
    }
    *blocks = *blocks + now4Blocks*4;

    return buf;
}

```

```

// mode: 0-> default
//      1-> '-l'
//      2-> '-lR'
int showDir(char* dirname, int mode) {
    if (mode == 2) {
        printf("%s:\n", dirname);
    }
    int blocks = 0;

    DIR* dir = opendir(dirname);
    struct dirent* dirDescribe;

```

```

struct stat st;
char nowDirnameBuf[1024];

/*for formatting*/
int indexForFileNames = 0; // index for allFileNames
int maxLengthOfFileName = 0;
char** allFileNames = NULL; // save all file's name to make sure format
if (mode == 0) {
    allFileNames = (char**)malloc(sizeof(char*) * 2048);
}

/*for keeping file info (kind of delay for us to get "total block")*/
int indexForInfos = 0;
char** allFileInfos = NULL;
if (mode != 0) {
    allFileInfos = (char**)malloc(sizeof(char*) * 2048);
}

/*for recursion*/
int indexForDirNames = 0;
char** allDirNames = NULL;
if (mode == 2) {
    allDirNames = (char**)malloc(sizeof(char*) * 2048);
}

while ((dirDescribe = readdir(dir)) != NULL) { // reading in loop

    strcpy(nowDirnameBuf, dirname);
    strcat(nowDirnameBuf, "/");
    strcat(nowDirnameBuf, dirDescribe->d_name);
    if (lstat(nowDirnameBuf, &st) {
        printf("error\n");
        return -1;
    }

    if (dirDescribe->d_name[0] != '.') { // hidden files off
        if (mode == 0) {
            char* nameBuf = (char*)malloc(sizeof(char) * 100);
            strcpy(nameBuf, dirDescribe->d_name);
            allFileNames[indexForFileNames] = nameBuf;
            indexForFileNames++;

            if (strlen(nameBuf) > maxLengthOfFileName)
                maxLengthOfFileName = strlen(nameBuf);
        }
    }
}

```



```

    } else {
        if (mode == 2) {
            if (S_ISDIR(st.st_mode)) { // is dir or not
                char* nameBuf = (char*)malloc(sizeof(char) * 100);
                strcpy(nameBuf, nowDirnameBuf);
                allDirNames[indexForDirNames] = nameBuf;
                indexForDirNames++;
            }
        }
        char* fileInfo = getFileInfo(&st, dirDescribe->d_name,
&blocks);

        allFileInfos[indexForInfos] = fileInfo;
        indexForInfos++;
    }
}

// output formatted information when mode is 0
if (mode == 0) {
    printFormatList(allFileNames,                maxLengthOfFileName,
indexForFileNames);
} else {
    printf("total:%d\n", blocks);
    printInfoList(allFileInfos, indexForInfos);

    if (mode == 2) {
        printf("\n");
        int i = 0;
        for (i = 0; i < indexForDirNames; i++) {
            showDir(allDirNames[i], 2); // enter recursion here
        }
    }
}

// free all heap space
if (mode == 0) {
    for (int i = 0; i < indexForFileNames; i++) {
        free(allFileNames[i]);
    }
    free(allFileNames);
} else {
    if (mode == 2) {
        for (int i = 0; i < indexForDirNames; i++) {

```

```

        free(allDirNames[i]);
    }
    free(allDirNames);
} else {
    for (int i = 0; i < indexForInfos; i++) {
        free(allFileInfos[i]);
    }
    free(allFileInfos);
}
}

closedir(dir);
return 0;
}

```

```

char* num2month(int num) {
    switch (num) {
        case 1:
            return "Jan";
        case 2:
            return "Feb";
        case 3:
            return "Mar";
        case 4:
            return "Apr";
        case 5:
            return "May";
        case 6:
            return "Jun";
        case 7:
            return "Jul";
        case 8:
            return "Aug";
        case 9:
            return "Sep";
        case 10:
            return "Oct";
        case 11:
            return "Nov";
        case 12:
            return "Dec";
        default:
            return "";
    }
}

```

```

}

void printFormatList(char** allFileNames,
                    int maxLengthOfFileName,
                    int indexForFileNames) {
    int i = 0;
    int num = WIDTH / (maxLengthOfFileName + 2);
    // a little trick here ( guess it wouldn't pass 99
    int temp = maxLengthOfFileName;
    char format[10];
    strcpy(format, "%-");
    if (temp > 9) {
        format[2] = temp / 10 + 48;
        format[3] = temp % 10 + 48;
        format[4] = '\\0';
    } else {
        format[2] = temp + 48;
        format[3] = '\\0';
    }
    strcat(format, "s  ");

    for (i = 0; i < indexForFileNames; i++) {
        printf(format, allFileNames[i]);

        if ((i + 1) % num == 0) {
            printf("\\n");
        }
    }
    printf("\\n");
}

void printInfoList(char** allFileInfos, int indexForInfos) {
    for (int i = 0; i < indexForInfos; i++) {
        printf("%s", allFileInfos[i]);
    }
}

```