Hello, Hello, World! World!

— Porting GO to ucore.

计 83 班 陈睿
计 83 班 方宇剑

陈睿

- Analysis of the program “Hello world”;

- Set up GDB debugging environment;

方宇剑

- Get to know the basic porting requirements;

陈睿

- Research on syscalls to be implemented;

方宇剑

- Confirm the maximum working set;
- Set up experiment environment and succeed in cross-compiling. Extract the os-dependent package;

陈睿

- Tool chain: a python script;
- runtime·exit1: exit the current os thread;

方宇剑

- Minimize working set;
- Tool chain: a shell script;
- runtime·setldt: fake ldt entries;

陈睿

- Fix some bug in runtime·write: stderr, etc;
- runtime· rt_sigaction: set to a direct return;

方宇剑

- runtime·write: repeated calls to putc();

陈睿

- runtime·mmap implementation;

- Disable environment constants and starting-up arguments;

方宇剑

- runtime·mmap implementation: the “with hint” version;

陈睿

- “Hello, world!”

方宇剑

- “Hello, world!”

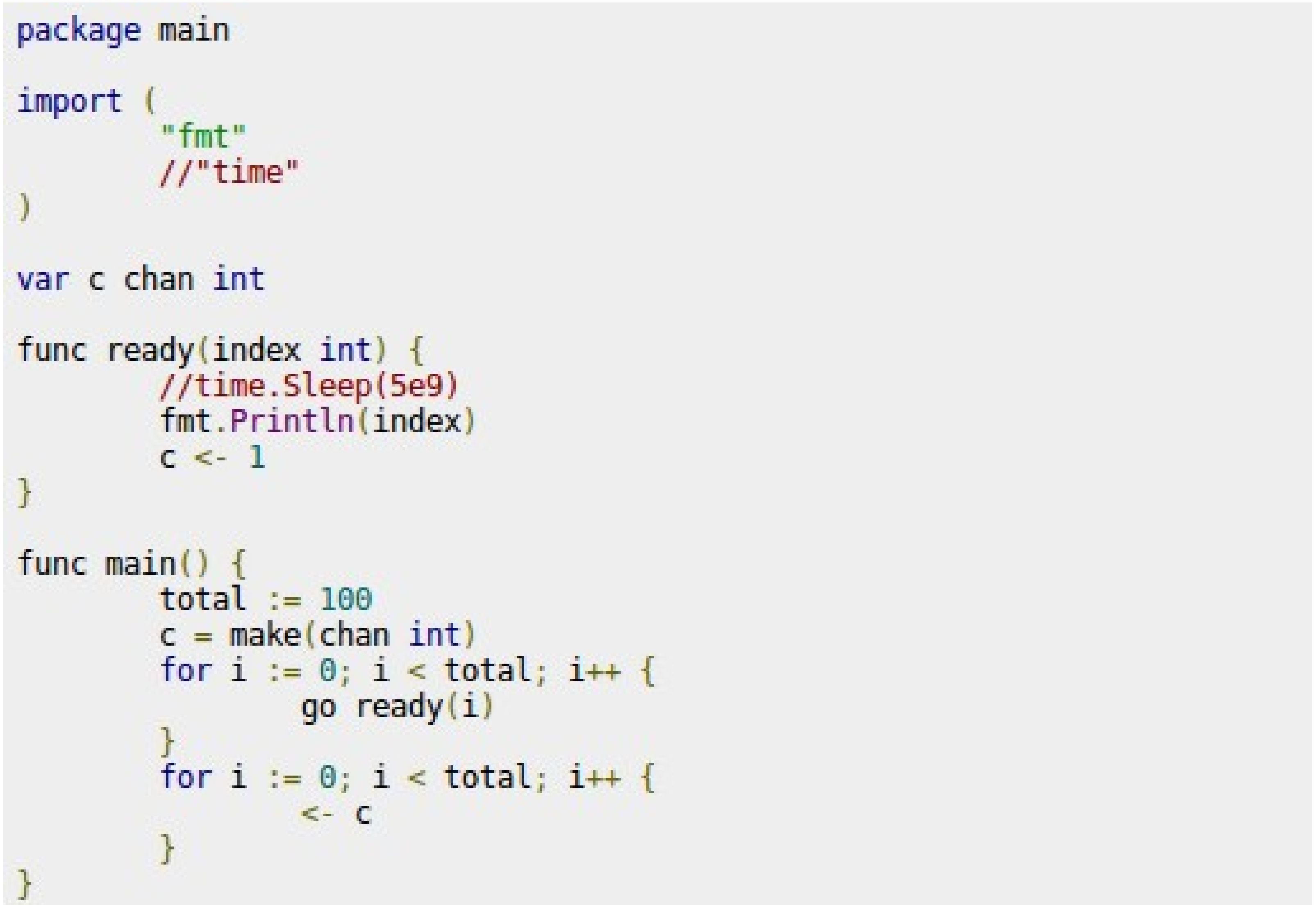
陈睿

- Read through Go specifications;

方宇剑

- Advanced hello world: using “fmt” package. “fmt” is os-independent -> “syscall” package;
- Read through Go specifications;
- peter.go: a program used for thread test;

peter.go



```go
package main

import (
        "fmt"
        //"time"
)

var c chan int

func ready(index int) {
        //time.Sleep(5e9)
        fmt.Println(index)
        c <- 1
}

func main() {
        total := 100
        c = make(chan int)
        for i := 0; i < total; i++ {
                go ready(i)
        }
        for i := 0; i < total; i++ {
                <- c
        }
}
```

陈睿

- Clone: return pid as tid;

方宇剑

- Research on goroutines in Linux;
- Clone: return pid as tid;

陈睿

- Attempts on semaphore: cause ucore to “reboot”;

方宇剑

- Updated version of the shell script: originates from the one used for a demo.

demo.sh

- -ng: disables the entire compiling of the Go compiler;
- -nre: disables the part-recompile of our packages (those lie in patch);
- -ntest: disables the compiling of the whole testsuit;
- Recursively compile all test cases;
- Automatically detect Makefile;
- Automatically generate testall.sh under each folder, which will be used in ucore;

陈睿

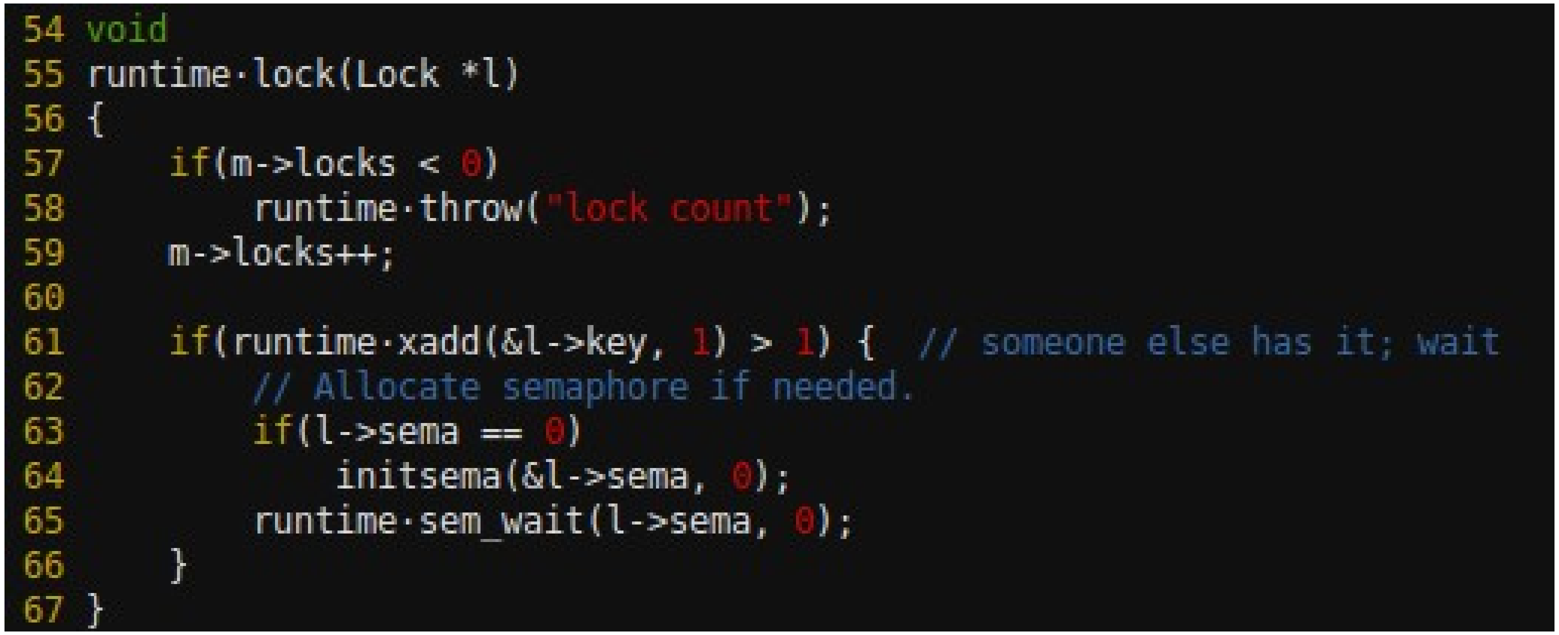
- Semaphore again: the nextm problem;

方宇剑

- Semaphore again: we are pretty sure the semaphore is set correctly, but still not working;

Lock/Unlock in Go

- The user-mode lock/unlock:



```
void
runtime·lock(Lock *l)
{
	if(m->locks < 0)
		runtime·throw("lock count");
	m->locks++;

	if(runtime·xadd(&l->key, 1) > 1) {  // someone else has it; wait
		// Allocate semaphore if needed.
		if(l->sema == 0)
			initsema(&l->sema, 0);
		runtime·sem_wait(l->sema, 0);
	}
}
```

陈睿

- Stuck in gcc 4.6 bootblock size;

方宇剑

- sleep: replace the former one using select;
- A user-leveled simulator: semtest5.c

陈睿

- The Println problem: it's not.

方宇剑

- Finally focus on the gs: ucore would not store/restore fs/gs! This would cause an unexpected running stream;
- Add fs/gs swap, problem solved;
- The Println problem: it's not.

陈睿

- Succeed in compiling ucore bootblock with gcc 4.6

方宇剑

- exit_group: exit all threads attached with the current os thread. Two attempts;

陈睿

- Testsuit;

方宇剑

- Testsuit;
- Environment constant: now contains "GOARCH=386", which is used for env.go;

Testsuit categories

- Compiler test: expected compiling error, tested and excluded;

- Linked test: provided for other test cases, normally requires Makefile;

- Generator test: used to generate another go file. Replaced;

- The rest: listed in testall.sh, should be tested in ucore. Panic on error (except for the panic test, see wiki);

To be done

- Signal!

  We have looked through the whole runtime package, which is the base of the GO world, and signal is the only one that has not been implemented;

Thanks!