Twisting Timing in your favour


Sie können
uns ruhig
abstrakte
Konzepte
vermitteln

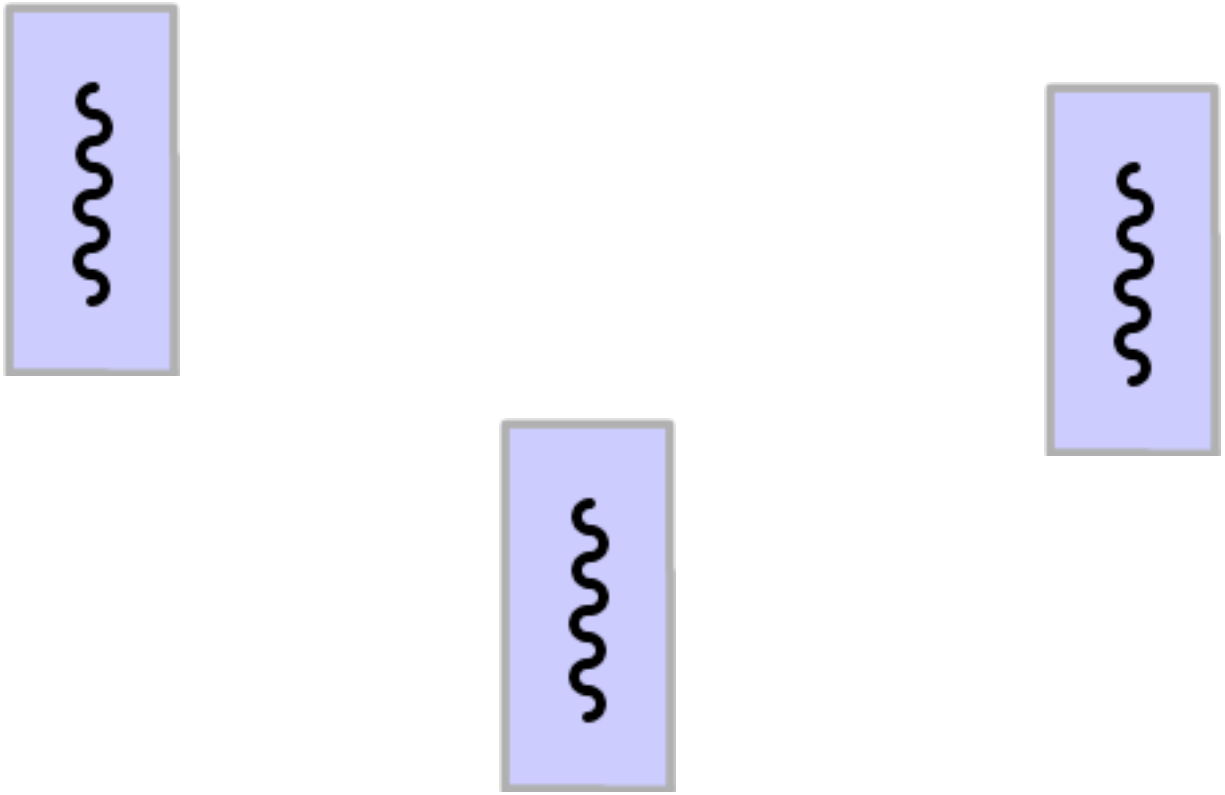
Outline

- Intro
- Common concurrency bugs – search & exploitation
- How to find concurrency bugs – general advice
- End note

What are concurrency bugs?

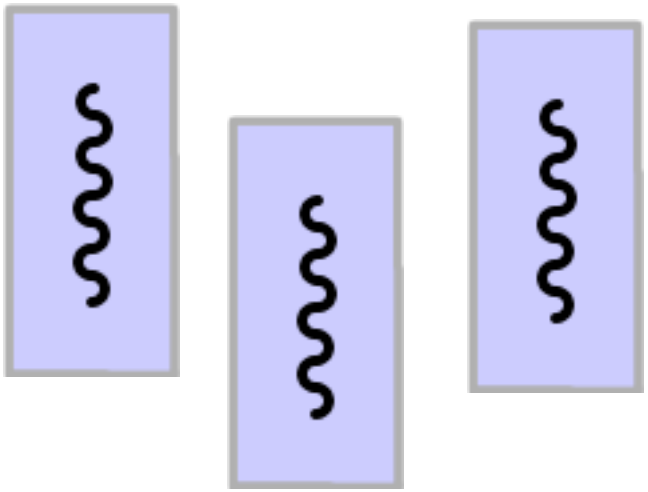
Where can concurrency bugs occur?

Multiple threads of execution…



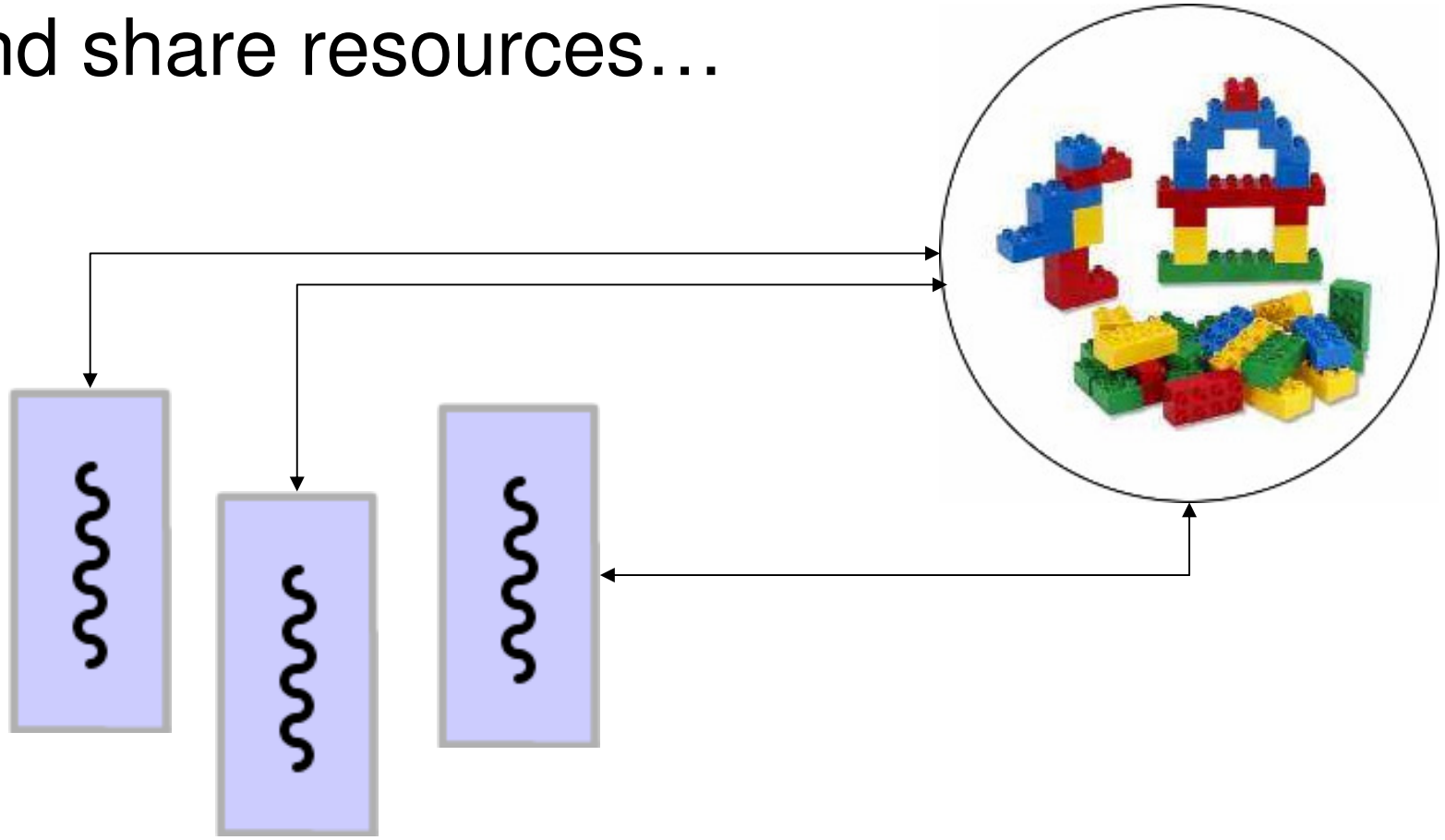
Where can concurrency bugs occur?

…that run simultaneously…



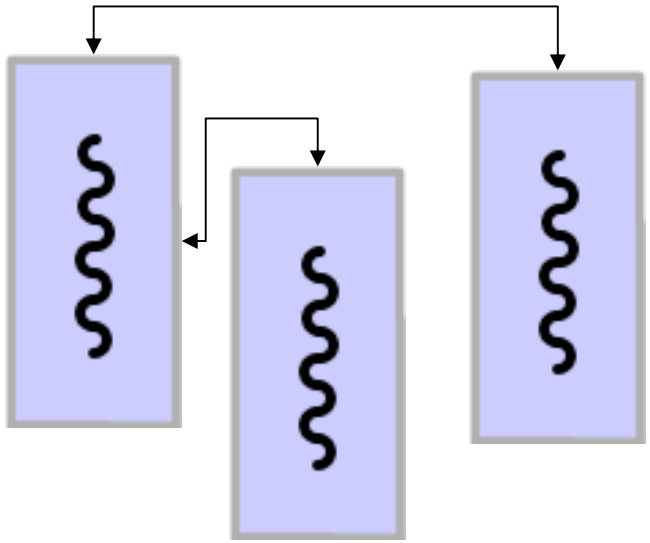
Where can concurrency bugs occur?

…and share resources…



Where can concurrency bugs occur?

…or interact in some other way…



Types of concurrency bugs

- Deadlocks
- Livelocks
- Mismatched communication
- False Sharing
- Race Conditions

Types of concurrency bugs

- Deadlocks
- Livelocks
- Mismatched communication
- False Sharing
- ***Race Conditions***

What is a Race Condition?

Vanilla Race Condition

```
#define CELLS 32
int rbf[CELLS];
int rbf_idx = 0;
uint32_t num_elems;
void rbf_store( int data ) {
    if (rbf_idx == CELLS)
        rbf_idx = 0;
    rbf[rbf_idx] = data;
    rbf_idx++;
    num_elems++; }
```

```
if (rbf_idx == CELLS)
        rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
```

31

32

0

1

```
if (rbf_idx == CELLS)
        rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
```

```
if (rbf_idx == CELLS)
        rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
```

31

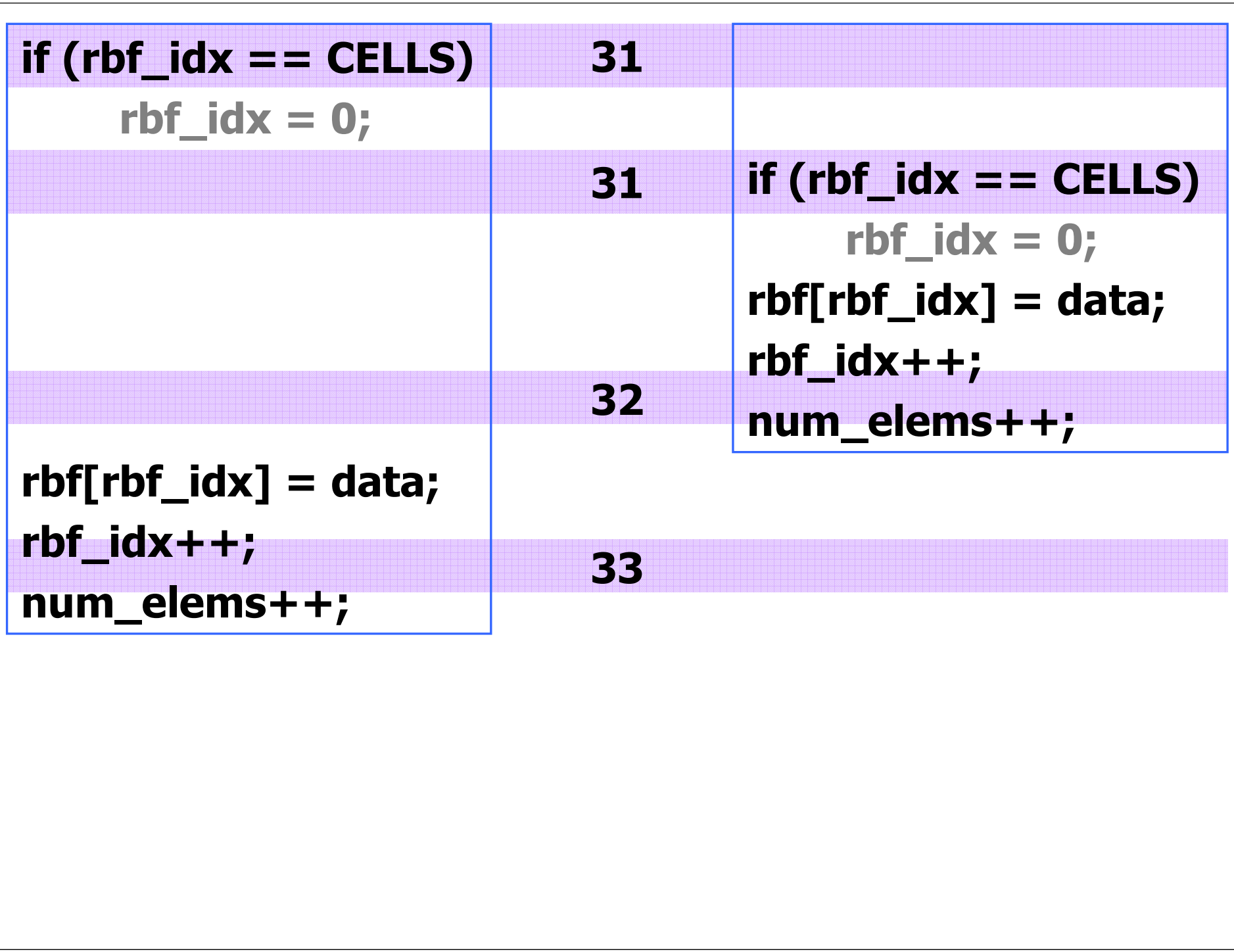
32

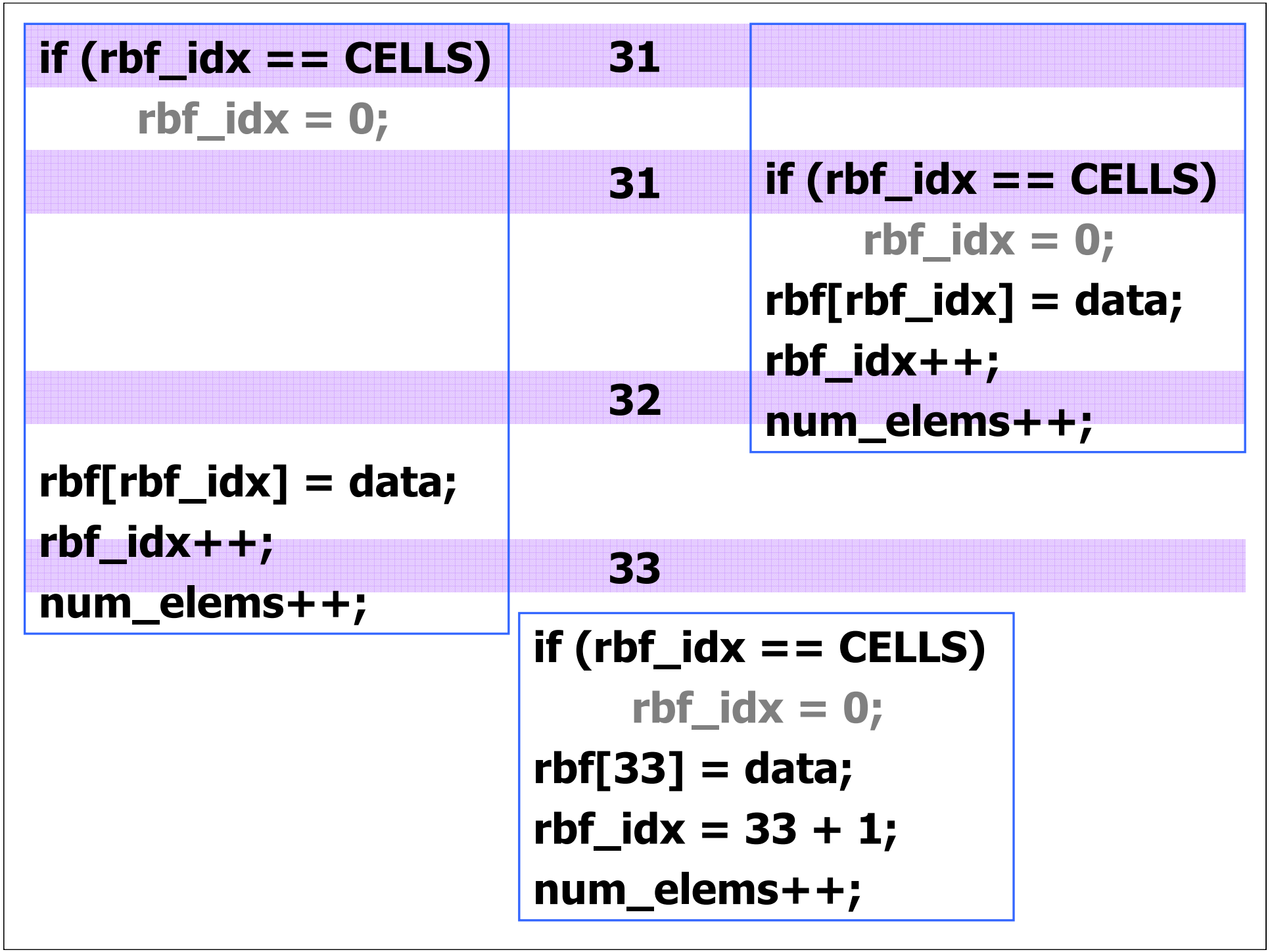
```
if (rbf_idx == CELLS)
        rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
```

0

1

```
num_elems++;
```


if (rbf_idx == CELLS)
rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
31
31
32
33
if (rbf_idx == CELLS)
rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;


if (rbf_idx == CELLS)
rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
31
31
if (rbf_idx == CELLS)
rbf_idx = 0;
rbf[rbf_idx] = data;
rbf_idx++;
num_elems++;
32
33
if (rbf_idx == CELLS)
rbf_idx = 0;
rbf[33] = data;
rbf_idx = 33 + 1;
num_elems++;

Why I like Race Conditions

Common Types of Race Conditions

- Forgotten lock
- File system races
- Signal races

Forgotten lock

Two types:

- No synchronization at all – or
- Inconsequent use

→ Vanilla Race Condition

Forgotten lock – finding it

- Synchronized access to all shared data and resources – in every single instance!
- Careful if several call layers are involved

File system race

```
$TempDir =
    $Config{'tmpdir'} . "logwatch." . $$ . "/";
if ( -d $TempDir ) {
    rmdir ($TempDir);
}
if ( -e $TempDir ) {
    unlink ($TempDir);
}
[...]
mkdir ($TempDir,0700);
```

```
$TempDir =
    $Config{'tmpdir'} . "logwatch." . $$ . "/";
if ( -d $TempDir ) {
    rmdir ($TempDir);
}
if ( -e $TempDir ) {
    unlink ($TempDir);
}
[...]
mkdir ($TempDir,0700);
```

```
$Command = $FileText . $FilterText . ">" .
    $TempDir . $LogFile;
[...]
`$Command`;
```

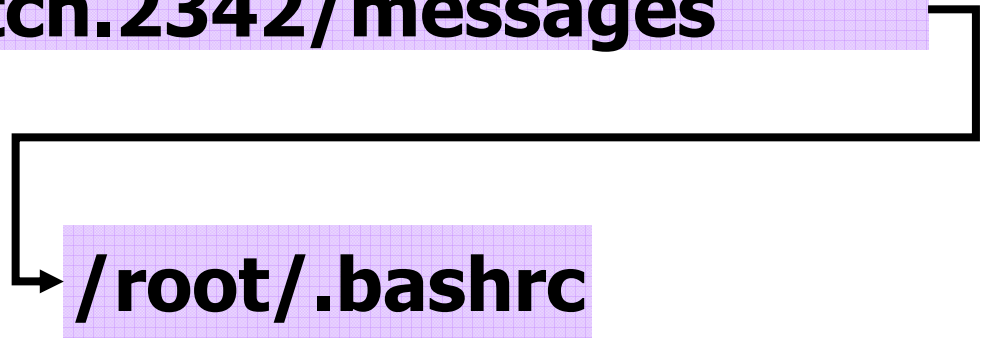
➔ $Command:

```
cat /var/log/secure|
   /etc/log.d/scripts/shared/applystddate
   >/tmp/logwatch.2342/secure
```

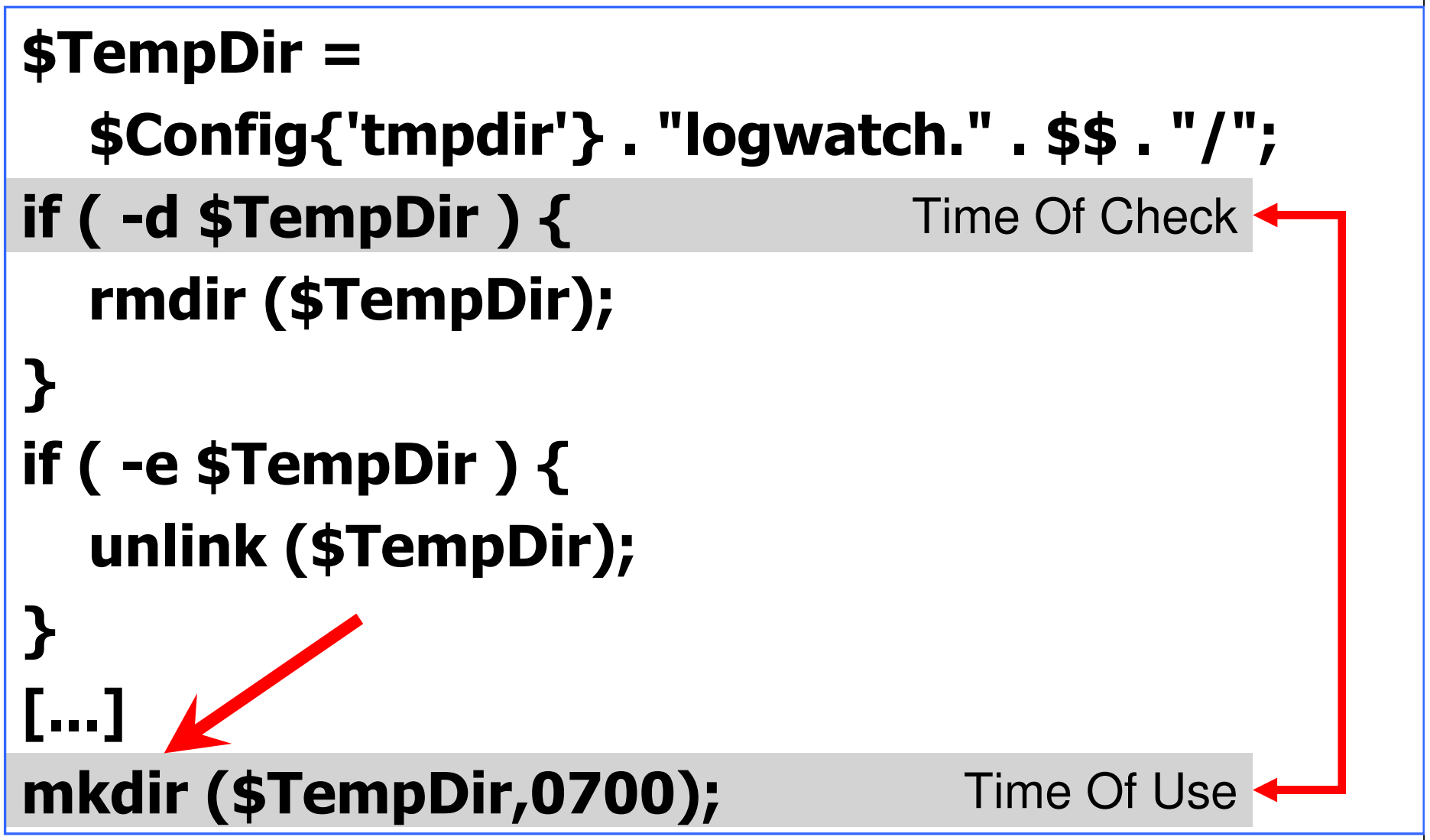
```
$Command = $FileText . $FilterText . ">" .
   $TempDir . $LogFile;
[...]
`$Command`;
```

➔ $Command:

```
cat /var/log/messages|
   /etc/log.d/scripts/shared/applystddate
   >/tmp/logwatch.2342/messages
```



```
$ cat /root/.bashrc
Jul   30 23:33:15 [...] <Some log messages>
Jul   30 23:33:15 [...] ; echo foo > /etc/passwd #
Jul   30 23:33:15 [...] <More log messages>
$
```


$TempDir =
$Config{'tmpdir'} . "logwatch." . $$. "/";
if (-d $TempDir) {
Time Of Check
rmdir ($TempDir);
}
if (-e $TempDir) {
unlink ($TempDir);
}
[...]
mkdir ($TempDir,0700);
Time Of Use

File system race – finding it

- Temp file handling is a good place to look
- Look for sequences of file operations that accept paths as arguments

Signal Race

- Calls to non-reentrant functions from signal handlers
- Reuse of non-reentrant signal handlers

Signal Race

MyServer – multithreaded httpd

```
void Sig_Quit(int signal)
{
        Server::getInstance()->logWriteln("Exiting...");
        sync();
        Server::getInstance()->stop();
}
```

```
        sig2.sa_handler = Sig_Quit;
        sigaction(SIGINT, &sig2,NULL); // catch ctrl-c
        sigaction(SIGTERM,&sig2,NULL); // catch the kill
```

Signal Race

Signal Race

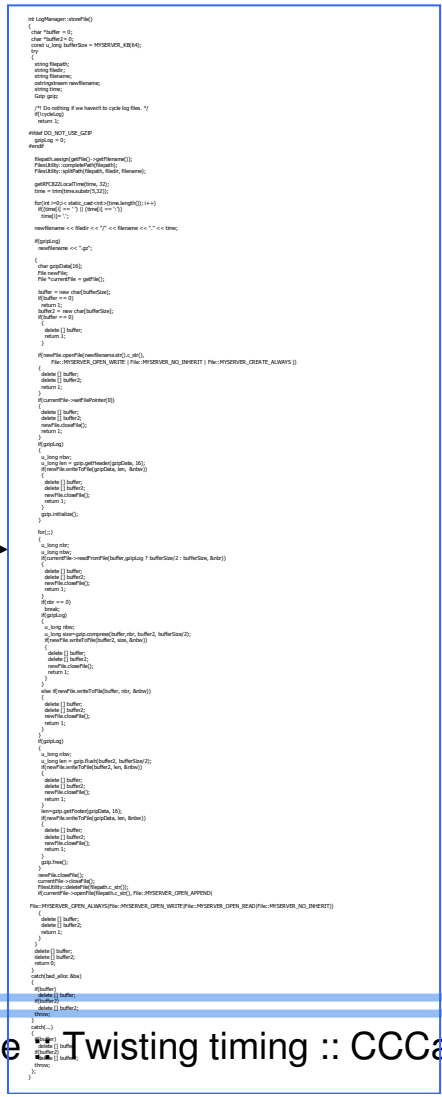
```
void Server::stop() {
        mustEndServer = 1;
}
```

Signal Race

int Server::logWriteln(const char* str)

```
int Server::logWriteln(const char* str)
{
  if(!str)
    return 0;

  /*
   *If the log receiver is not the console output a timestamp.
   */
  if(logManager.getType() != LogManager::TYPE_CONSOLE)
  {
    char time[38];
    int len;
    time[0] = '[';
    getRFC822GMTTime(&time[1], 32);
    len = strlen(time);
    time[len + 0] = ']';
    time[len + 1] = ' ';
    time[len + 2] = '-';
    time[len + 3] = '-';
    time[len + 4] = ' ';
    time[len + 5] = '\0';
    if(logManager.write(time))
      return 1;
  }
  return logManager.writeln((char*)str);
}
```

```
int LogManager::write(const char *str, int len)
{
  int ret;
  if(type == TYPE_CONSOLE)
  {
    cout << str;
  }
  else if(type == TYPE_FILE)
  {
    u_long nbw;

    /*!
     *File was not loaded correctly.
     */
    if(!loaded)
    {
      return 1;
    }

    /*!
     *We reached the max file size.
     *Don't use this limitation if maxSize is equal to zero.
     */
    if(maxSize && (file.getFileSize() > maxSize))
    {
      if(storeFile())
        return 1;
    }

    /*!
     *If the len specified is equal to zero write the string as
     *a null character terminated one.
     */
    ret = file.writeToFile(str, len ? len : (u_long)strlen(str), &nbw);
    return ret;
  }
  return 0;
}
```



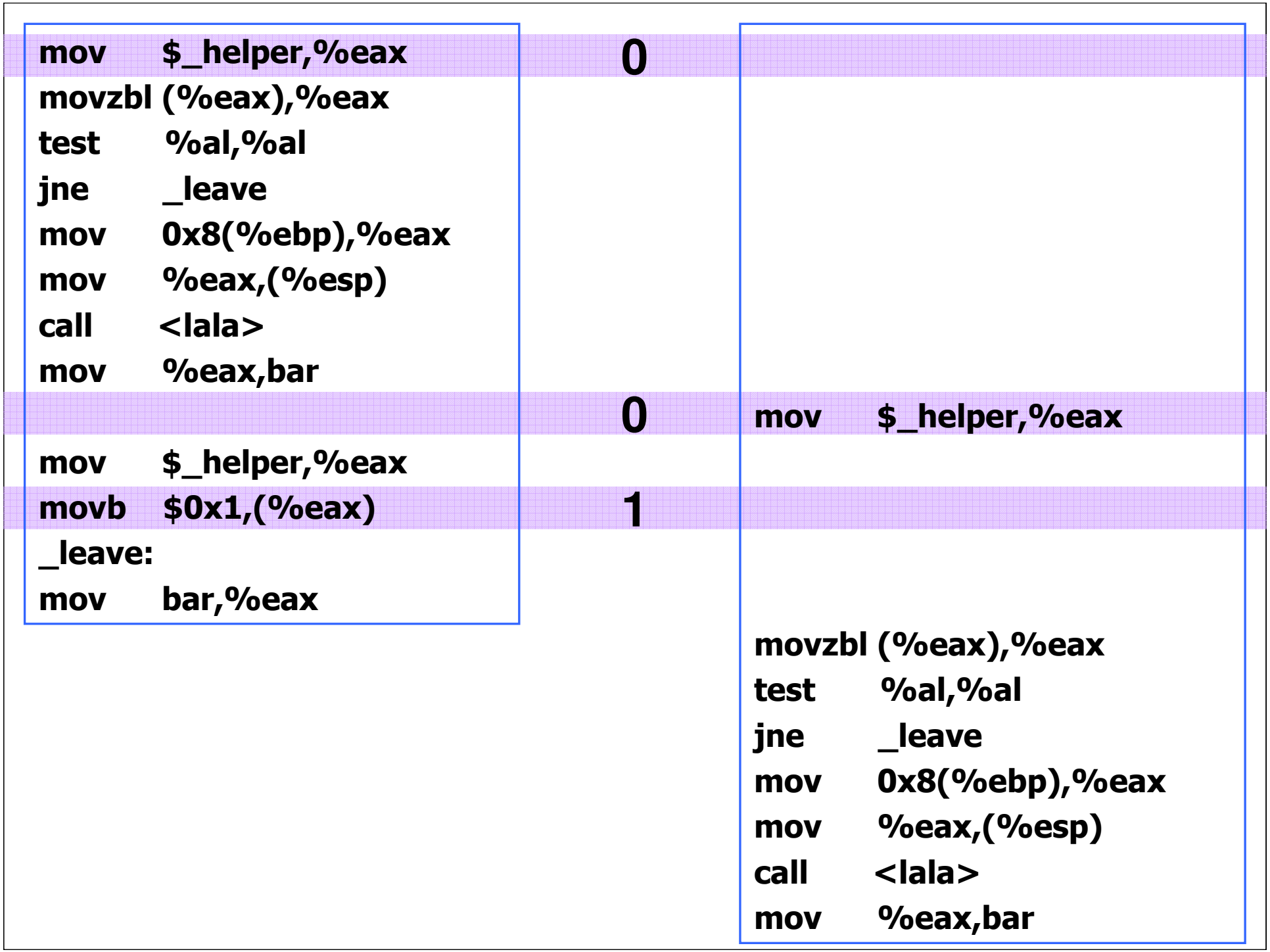
Signal Race – finding it

- Find sigaction()
- Find functions passed to sigaction()
- Inspect them

Local static initialization in g++

```
int foo()
{
    static int bar = lala();

    // More fancy code here
    // ...
}
```

```
mov     $_helper,%eax          Load helper
movzbl  (%eax),%eax
test    %al,%al                Test helper
jne     _leave
mov     0x8(%ebp),%eax
mov     %eax,(%esp)
call    <lala>
mov     %eax,bar               Initialize local static
mov     $_helper,%eax          Load helper
movb    $0x1,(%eax)            Lock down helper
_leave:
mov     bar,%eax
```


mov $_helper,%eax
movzbl (%eax),%eax
test %al,%al
jne _leave
mov 0x8(%ebp),%eax
mov %eax,(%esp)
call <lala>
mov %eax,bar
mov $_helper,%eax
movb $0x1,(%eax)
_leave:
mov bar,%eax
0
0
1
mov $_helper,%eax
movzbl (%eax),%eax
test %al,%al
jne _leave
mov 0x8(%ebp),%eax
mov %eax,(%esp)
call <lala>
mov %eax,bar

More educational examples

- Dietlibc (this is _not_ triggerable!)

```
#define BUF_SIZE 2048
#define MAX_LOGTAG 1000
static char LogTag[MAX_LOGTAG];
void __libc_openlog(const char *ident, int option, int facility) {
  if (ident) {
    strncpy(LogTag,ident,MAX_LOGTAG);
    LogTag[MAX_LOGTAG-1]=0;
  } ... }

void __libc_vsyslog(int priority, const char *format, va_list arg_ptr) {
 char buffer[BUF_SIZE];
  int headerlen =
    snprintf(buffer, 130, "<%d>%s %s: ", priority, time_buf, LogTag);
  [...]
  buflen =
    vsnprintf(buffer+headerlen, BUF_SIZE - headerlen, format, arg_ptr);
}
```

Bizarre Race Conditions

MyServer, server.cpp, line 679, in
Server::terminate (not a deconstructor):

```
threadsMutex->lock();
threads.clear();
threadsMutex->unlock();
delete threadsMutex;
```

General Advice: Finding RCs

- Look out for resources that are not obviously shared
  - Classic: libc function calls that use static variables internally
- Look out for constructs that indicate Singletons: instanceOf and friends

Closing words

Software is *not* safe just because…

- … it's not multithreaded
- … it's not written in C/C++
- … it's written in Erlang or friends