15-410
"What could possibly go wrong?"

"Paradise Lost"
Feb. 9, 2024

Dave Eckhardt

Outline

When to use `if()` vs. `while()`

Consider the lowly worker thread

```
/* note: not a thrgrp_*() worker thread */
void
worker(void *ignored)
{
  workitem *work;
  while (work = find_work())
      perform(work);
  thr_exit((void *) 0);
}
```

Consider the lowly worker thread

```
/* note: not a thrgrp_*() worker thread */
void
worker(void *ignored)
{
  workitem *work;
  while (work = find_work())
      perform(work);
  thr_exit((void *) 0);
}
```

But a funny thing happens when this code is run...

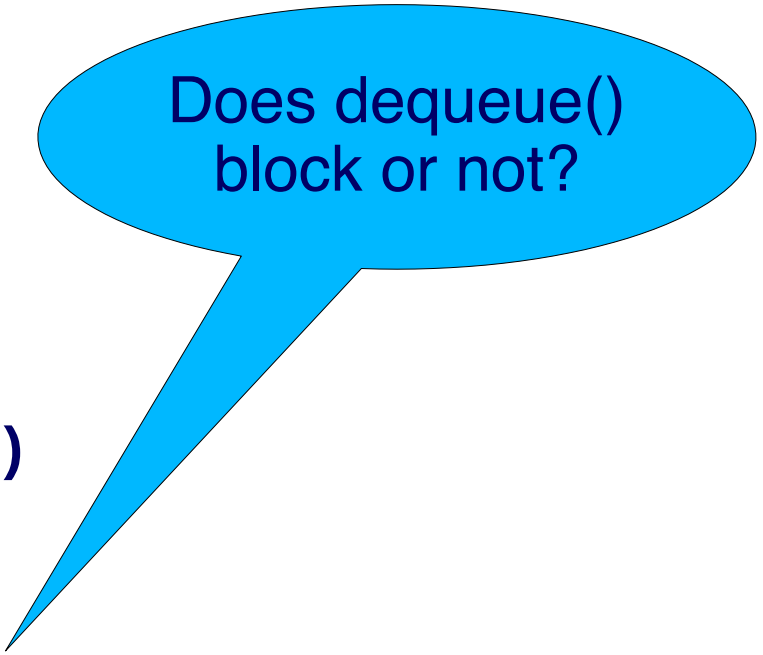
- Better look at `find_work()`?

What's Wrong With This Picture?

```
workitem *
find_work(void)
{
  workitem *w;
  mutex_lock(&m);
  if (going_out_of_business)
      w = (workitem *) 0;
  else
      w = (workitem *) dequeue(q);
  mutex_unlock(&m);
  return (w);
}
```

What's Wrong With This Picture?

```
workitem *
find_work(void)
{
  workitem *w;
  mutex_lock(&m);
  if (going_out_of_business)
      w = (workitem *) 0;
  else
      w = (workitem *) dequeue(q);
  mutex_unlock(&m);
  return (w);
}
```



Better?

```
mutex_lock(&m);
if (going_out_of_business) {
      w = (workitem *) 0;
} else {
  if (!(w = (workitem *) dequeue(q))) {
      cond_wait(&new_work, &m);
      w = (workitem *) dequeue(queue);
  }
}
mutex_unlock(&m);
return (w);
```

Better?

```
mutex_lock(&m);
if (going_out_of_business) {
      w = (workitem *) 0;
} else {
  if (!(w = (workitem *) dequeue(q))) {
      cond_wait(&new_work, &m);
      w = (workitem *) dequeue(queue);
  }
}
mutex_unlock(&m);
return (w);
```

But the funny thing *still* happens... less...?

What We Hope For

find_work()	queue_work()
`mutex_lock(&m);`	
`if (!..dequeue(..))`	
`cond_wait(&new, &m);`	
	`mutex_lock(&m);`
	`enqueue(..);`
	`cond_signal(&new);`
	`mutex_unlock(&m);`
`w = dequeue(..);`	
`mutex_unlock(&m);`	

What Went Wrong?

What went wrong?

What Went Wrong?

What went wrong?

- **Nothing!**

What Went Wrong?

What went wrong?

- **Nothing!**

But what if there is *an evil third thread*?

What About the “Evil Third Thread”?

```
mutex_lock(&m);
if (going_out_of_business) {
      w = (workitem *) 0;
} else {
  if (!(w = (workitem *) dequeue(q))) {
      cond_wait(&new_work, &m);
      w = (workitem *) dequeue(queue);
  }
}
mutex_unlock(&m);
return (w);
```

Not Exactly What We Hope For

find_work()	queue_work()	find_work()
lock(&m);		
if (!..deq(..))		
cwait(&new, &m);		
	lock(&m);	
	enqueue(...);	
	csignal(&new);	
	unlock(&m);	
		lock(&m);
		if (!..deq(..))
		unlock(&m);
w = deq(..);		return(w);
unlock(&m);		
return (0);		

Have We Seen This Before?

What went wrong?

- Protected world state wasn't ready for us
- We blocked
- Somebody prepared the world for us to run
- We ran
  - We *assumed* nobody else had run
  - We *assumed* the world state was still ready for us

When have we seen this "happiness revocation"?

To “if()” Or Not To “if()”?

```
mutex_lock(&m);
if (going_out_of_business) {
    w = (workitem *) 0;
} else {
    while (!(w = (workitem *) dequeue(q)))
        cond_wait(&new_work, &m);
}
mutex_unlock(&m);
return (w);
/* XXX still wrong! - rewrite after class */
```

Summary

if() vs. while()

- If somebody can revoke your happiness, you'd better check

Related Work

TOCTTOU

- ?

Related Work

TOCTTOU

"Toucan at Whipsnade Zoo", William Warby, 2012-05-06, CC-BY

Related Work

TOCTTOU

- "Time of Check to Time of Use"
  - A standard "bug class"
  - Isn't that what we have here?

Related Work

TOCTTOU

- "Time of Check to Time of Use"
  - A standard "bug class"
  - Isn't that what we have here?
- "Correct, but wrong"

Related Work

TOCTTOU

- "Time of Check to Time of Use"
  - A standard "bug class"
  - Isn't that what we have here?
- "Correct, but wrong"
  - Many people think TOCCTOU bugs are always security bugs
  - Fundamentally, we expect the revoked condition to become unrevoked again (soon!)
  - Unlike the general case, this can be fixed in less than a line of code!