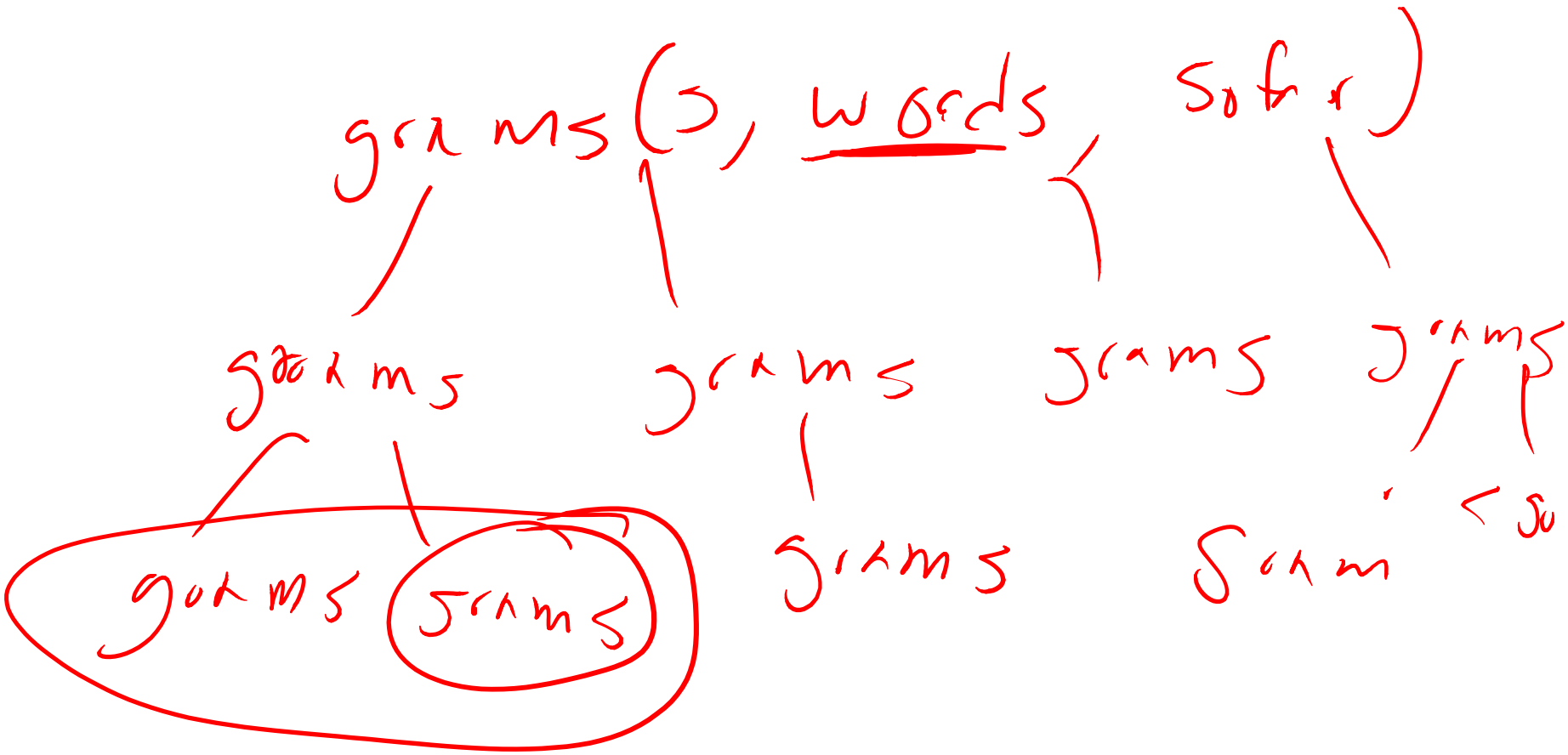
CS 150: SAFELY Sharing Between Processes

Cynthia Taylor
Oberlin College
April 21st 2014

Lab: Common Mistake

```
def grams(s, words, sofar):
  if len(s) == 0:
     print(" ".join(sofar))
  for w in words:
     ....
     sofar = sofar + [w]
     grams(remains,words,sofar)
```


grams(s, words, sofar)
grams
grams
grams
grams
grams
grams
grams
grams
sofar

Last class: Shared Memory

- Used RawValue to let multiple processes use the same variable

- But this could give us inconsistent results!

Locks

- Shared between processes
- Only allows one process to be in a section of “locked” code at a time.

Syntax

```
l = Lock()

l.acquire()
#dangerous code
l.release()
```

only one process can hold lock

What is the BEST version of this code?

```
def func(r,lock):
    lock.acquire()
    for i in range(10):
        r.value = r.value + 1
        time.sleep(.05)
    lock.release()
```

A

```
def func(r,lock):
    for i in range(10):
        lock.acquire()
        r.value = r.value + 1
        lock.release()
        time.sleep(.05)
```

B

```
def func(r,lock):
    for i in range(10):
        r.value = r.value + 1
        lock.acquire()
        time.sleep(.05)
        lock.release()
```

C

D. Multiple options are equally good

E. I don't know

Demo

Will this code give us consistent results?

```
def func(r):
    lock = Lock()
    for i in range(10):
        lock.acquire()
        r.value = r.value + 1
        lock.release()
        time.sleep(.05)
```

A. Yes

B. No

C. I don't know

need shared lock

l = Lock()

p = Process(target=func, args=(r, l))

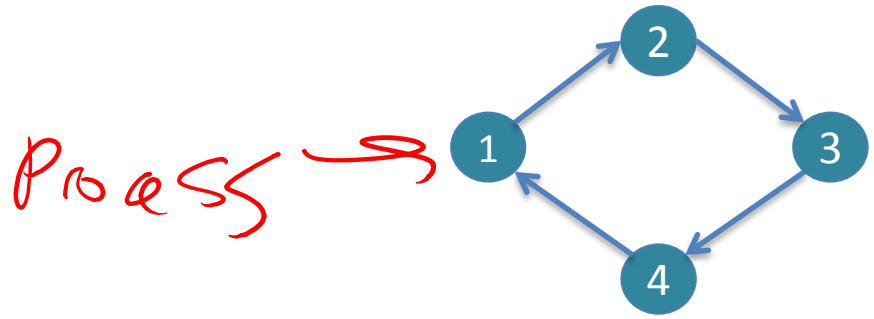
What will happen if we run this code?

```
def func(r,lock):
    for i in range(10):
        lock.acquire()
        r.value = r.value + 1
        time.sleep(.0001)
r = RawValue("i",0)
lock = Lock()
for i in range(10):
    p = Process(target=func, args=(r,lock))
    p.start()
```

A. Everything will be okay
B. It will cause an error
C. It will print once and then stop
D. Something else
E. I don't know

always all release!

Deadlock



p1 aquires L1
p1 waits L2
p2 aquired L2
p2 waiting L1

- Occurs when we have a cycle of processes all waiting on each other
- No process can make progress
- Program will hang forever

Which of these could cause deadlock?

```
def func(l1,l2):
    l1.acquire()
    l2.acquire()
    #code
    l2.release()
    l1.release()

def func2(l1,l2):
    l2.acquire()
    l1.acquire()
    #code
    l1.release()
    l2.release()
```

A

```
def func(l1,l2):
    l1.acquire()
    #code
    l1.release()
    l2.acquire()
    #code
    l2.release()

def func2(l1,l2):
    l2.acquire()
    #code
    l2.release()
    l1.acquire()
    #code
    l1.release()
```

B

```
def func(l1,l2):
    l1.acquire()
    l2.acquire()
    #code
    l1.release()
    l2.release()

def func2(l1,l2):
    l2.acquire()
    l1.acquire()
    #code
    l2.release()
    l1.release()
```

C

D. More than one of the above

E. I don't know

Avoiding Deadlock

- Never call acquire() without calling release()
- Be VERY careful using multiple locks

Bank Account

Next Time

- Critter Tournament!! Fabulous Prizes
- Lab 11 – Tuesday at 10 pm

Last Lab

Practice Final on Blackboard