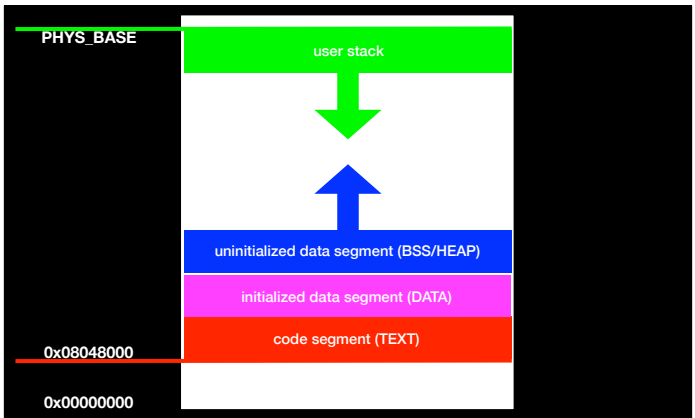
Threads

COS 450 - Fall 2018

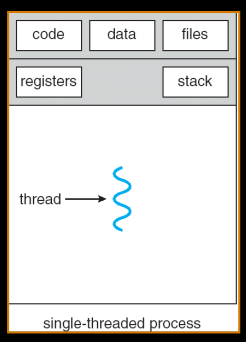
1



Pintos Process Memory

2

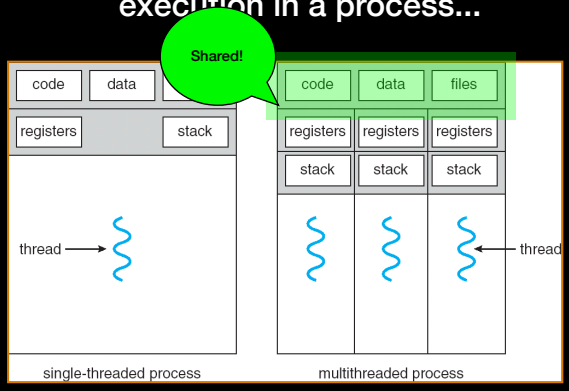
So far we have looked at **single** paths of execution in a process...



3

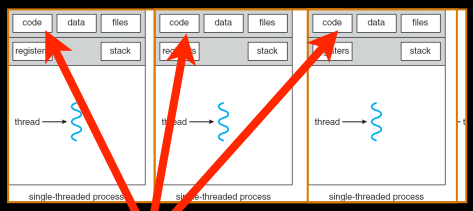
4

Threads are **multiple** paths of execution in a process...



5

so why use threads?
aren't multiple processes enough?



3x the code, data, file, IO...
in this example

6

Benefits

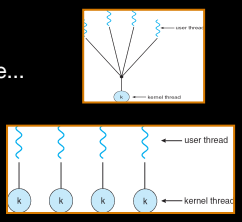
- Resource Sharing
- Economy
- Responsiveness
- Utilize Multi-processors

Thread Models

7

How can we best design threads to work?

in user-space...

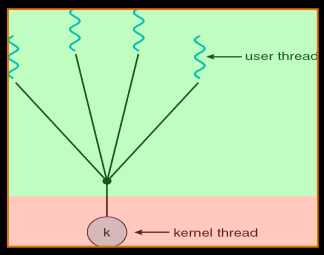


Many to One

8

one **kernel** object for each **process**, threads **share**.

they all take turns executing.



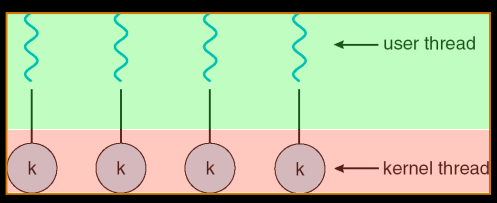
...if one blocks they all block.

One to One

9

one **kernel** object for each **thread**

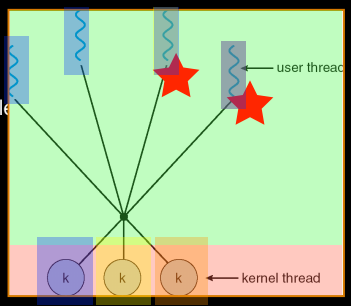
...threads are nearly processes.



10

Many to Many

user threads multiple

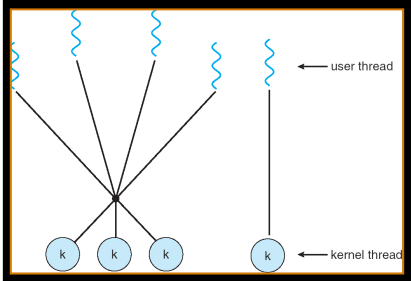


if one blocks...

...assign a new kernel object to wait on

11

Mixed Model



we can also allow a **thread** to bind itself to a **kernel object**.

where is this useful?

12

Problems?

how do **fork()** and **exec()** work?

how do you **cancel** a thread?

where do **signals** (interrupts) go?

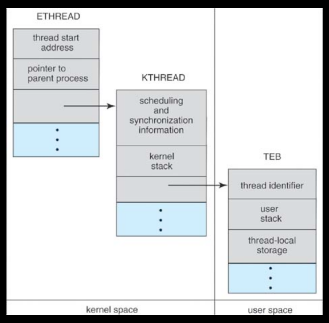
what about **priority**?

Win32

13

Uses the
One-to-One Model

...with fibers (many to one)



Linux

14

clone() instead of fork()

process = thread = task

One-to-One Model

Questions

15

- When does multithreading not help?
- What problems is it great for?
- How could you decrease the thread creation time (thread pools)?

End of Threads

for now...