Process: a running program ← typically protected

Thread: a with a process ← shared memory
with a separate stack

Race conditions

\[
\begin{aligned}
Th1 & : & X & = X + 1 \\
Th2 & : & Y & = y + 1 & \text{OK} \\
& & X & = X + 1 & \text{problem}
\end{aligned}
\]

\[
\begin{aligned}
X & = X - 2
\end{aligned}
\]

Th1
MOV X, RI
ADD 1, RI conflict: -
MOV RI, X get the wrong result

Th2
MOV X, RI

Critical Section

POSIX Threads

pthread_t m;
pthread_mutex_lock (m);

Java:
synchronized (obj)

pthread_mutex_unlock(m)
thread 1()  \rightarrow \{ \} \rightarrow \text{thread 2}
\xrightarrow{X()}\rightarrow A\rightarrow b(\cdot)
\xrightarrow{Y()}\rightarrow X\rightarrow B\rightarrow c(\cdot)
\xrightarrow{Z()}\rightarrow Y\rightarrow C\rightarrow d(\cdot)
\xrightarrow{Z()}\rightarrow Z\rightarrow P

cop multi-threading: not much
pre-emptive MT disable interrupts
multi-core: spin lock

\begin{array}{c}
\text{CPU1} \\
1 \\
\swarrow \text{lock} \\
0 \\
\swarrow \text{SWAP} \\
1 \\
\swarrow \text{SWAP} \\
0 \\
\swarrow \\
0
\end{array}
\begin{array}{c}
\text{CPU2} \\
1 \\
\swarrow \\
0 \\
\swarrow \text{SWAP} \\
0 \\
\swarrow \\
0
\end{array}
CPU 1  lock  CPU 2

1 0 1

thread 1  thread 2

Control block

mutex

spin lock

owner

queue

mutex.lock:

if trylock(spinlock)

spinlock(lock)

if owner = null

owner = me

else

add myself to queue

release(lock)

if owner != me

sleep

th3

th4