

ITP30002 Operating System

## Homework 4

# Runtime Deadlock Detector

# Overview

- Develop a runtime monitoring tool for checking resource deadlock of Pthread mutexes using runtime interpositioning
  - Target to detect cyclic deadlocks of mutexes in multithreaded C programs using Pthread
  - Construct a shared library that overrides Pthread APIs
- Demonstrate with example programs that your tool effectively detects occurrence of deadlocks

# Runtime Deadlock Detector (1/2)

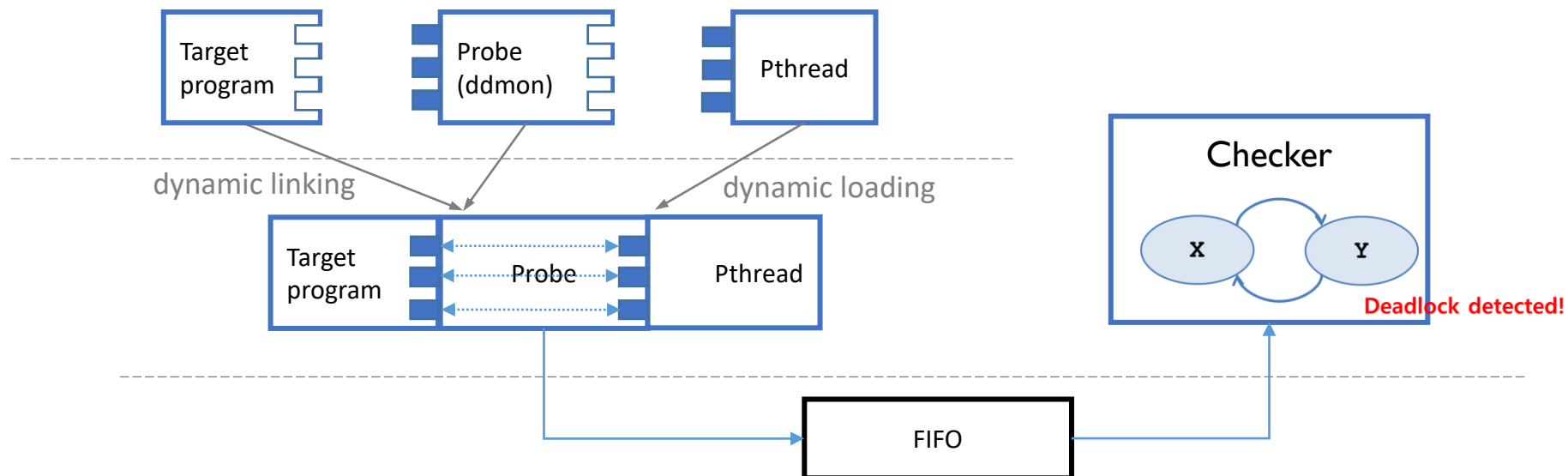
- Construct a cyclic deadlock monitor *ddmon*
  - *ddmon* should implement the following cyclic deadlock detection algorithm for Pthread mutex

## Cyclic deadlock monitoring algorithm (e.g. *LockDep*)

- Monitor lock acquires and releases in runtime
- Lock graph  $(N, E_N)$ 
  - Create a node  $n_X$  when a thread acquires lock  $X$
  - Create an edge  $(n_X, n_Y)$  when a thread acquires lock  $Y$  while holding lock  $X$
  - Remove  $(*, n_X)$  when a thread releases  $X$ . Remove  $n_X$  when a thread releases  $X$  and no other threads had acquired  $X$
  - Report deadlock when the graph has any cycle

# Runtime Deadlock Detector (1/2)

- ddmmon should consist of the probe part and the checker part
  - **Probe part:** should be implemented as a dynamic library `ddmon.so` (`ddmon.c`) that overrides certain Pthread APIs. This module should be linked with a target program to emit the runtime information for checking deadlock in an execution
  - **Checker part:** should be implemented as an independent program `ddchck.c` that receives the emitted information from the probe for checking cyclic deadlocks
  - Probe transfers runtime information to Checker via FIFO



# Runtime Deadlock Detector (2/2)

- Checker should alert about a deadlock occurrence when a cycle is constructed at the lock graph
- An alert must print out the identifiers of the threads that are involved in the deadlock, and also memory addresses of the mutexes involved in the deadlock
- **Extra point.** an alert shows the source code line numbers where one or more mutexes involved in a deadlock are acquired
  - Use `backtrace()` with `addr2line`
  - Use `popen()` when it uses `addr2line`

[https://www.gnu.org/software/libc/manual/html\\_node/Pipe-to-a-Subprocess.html](https://www.gnu.org/software/libc/manual/html_node/Pipe-to-a-Subprocess.html)

# Assumptions

- Assume that a target program creates no more than 5 threads and no more than 10 mutexes
- Assume that the target program and ddchck are located at the same directory
  - Create and use a FIFO “.ddtrace” at the same directory
  - Your program may assume that .ddtrace is created before program execution
- ddchck receives a target program object as a command-line input to obtain the target program source code information
  - E.g. when the target program is a.out,  
\$ ddchck a.out

# Write Up and Video Demo

- Your write-up should detail how ddmon implements the cyclic deadlock detection
  - describe which Pthread APIs are overridden, and how
  - how to handle synchronization in constructing the tool
  - the designed protocol of Probe and Checker communication
  - etc.
- You must write example programs to demonstrate ddmon accurately detects deadlocks
  - one of them must involve a deadlock with more than 2 threads
- Submit the source code files as well as the build scripts for the techniques and your example programs
- Take a video clip of the demonstration (less than 3 minutes)

# Submission

- Deadline: 9 PM, 9 June (Tue)
  - no late submission will be accepted
- Your submission must include the followings:
  - Write-up: up to 3 pages in the given template
  - URL of your video demo (e.g., YouTube)
    - put the URL in your write-up
  - All related source code files
- How to submit
  - upload your files to a homework repository in Hisnet



# Evaluation

- Points

- Technical soundness      40% (+20% for the extra-point task)
- Demonstration              30%
- Presentation                20%
- Discussion                  10%

- Note

- Evaluation will be primary on your write-up and video demo
- TAs will test the submitted files on the peace server