PI-fråga: Då en pipe skapas inom en process som är flertrådad så kan flera trådar läsa från pipen och skriva till pipen utan att behöva stänga några läs eller skrivändar, trådarna kör ju samma adressrymd och deskriptorer dubbleras inte av någon `fork()` eftersom vi inte kör någon `fork()`. Antag nu att vi har skapat pipen `p`. Då kan `p` användas istället för en mutex på följande sätt:

a) Varje tråd tilldelas ett unikt heltal, kalla detta för tråd-ID. Då en tråd vill låsa `p` skriver tråden sitt tråd-id till pipen två gånger och läser från pipen en gång. Om den då läser sitt eget tråd-id kan den vara vara säker på att låsningen lyckats och därefter anser den att den äger resursen i fråga. Att trådens id ligger i pipen är nu ett tecken på att tråden låst mutexen.

b) Om den läser någon annans tråd-id, kalla det för X, betyder det att någon annan tråd (med id=X) hunnit före att skriva in sitt tråd-id. Det betyder att tråden inte lyckats låsa mutexen. Då läser tråden ända tills den hittar sitt id och läser bort dessa och skriver tillbaka X för att återställa pipen i det läge som den var, det vill säga X ska finnas där och indikera att tråden med id X har låst mutexen.

A. Ovanstående fungerar.

B. Ovanstående fungerar inte.

C. Ovanstående handlar om att slippa en mutex, och det fungerar om man lägger till en koordinationstråd som ser till att inga kapplöpningar uppstår kring läsningar och skrivningar med pipen.

PI-fråga: Då en pipe skapas inom en process som är flertrådad så kan flera trådar läsa från pipen och skriva till pipen utan att behöva stänga några läs eller skrivändar, trådarna kör ju samma adressrymd och deskriptorer dubbleras inte av någon `fork()` eftersom vi inte kör någon `fork()`. Antag nu att vi har skapat pipen `p`. Då kan `p` användas istället för en mutex på följande sätt:

a) Att resursen som mutexen/pipen vaktar är ledig indikeras av att det ligger ett tal (vilket som helst) i pipen. Då en tråd vill låsa mutexen/pipen gör tråden en `read` på pipen. Eftersom det finns något att läsa så blockeras inte tråden utan kan fortsätta `read` är alltså samma som *lock* (*Get_mutex*/`pthread_mutex_lock()`). En läsning resulterar också i att pipen är tom vilket kommer att förorsaka senare trådar att vänta.

b) Då en tråd vill låsa upp mutexen/pipen så skriver tråden ett tal (vilket som helst) i pipen, då finns något att läsa. Så länge det inte finns något att läsa så blir trådar som försöker läsa blockerad, men när nu ett tal är skrivet så när en tråd försöker läsa så går det och detta fungerar alltså som en upplåsning `write` är alltså här detsamma som *unlock* (*Release_mutex*/`pthread_mutex_unlock()`).

A. Ovanstående fungerar.

B. Ovanstående fungerar inte.

C. Man kan få ovanstående att fungera genom att låta pipen vaktas av en mutex. Det är mycket vunnet med detta.