ບົດທີ 4

ແນວຄິດກ່ຽວກັບ Thread (Thread Concepts)

ເນື້ອໃນຫຍໍ້

- ສະເໜີກ່ຽວກັບ Thread (Introduction)
- ນິຍາມຂອງ Thread (Definition of Thread)
- ແຮງບັນດານໃຈໃນການສ້າງ Thread (Motivation for Thread)
- ສະຖານະພາບຂອງ Thread (Thread State)
- ການດຳເນີນການກ່ຽວກັບ Thread (Thread Operations)
- Threading Model
- ສິ່ງທີ່ຄວນພິຈາລະນາໃນການສ້າງ Thread
- POSIX ແລະ Pthreads

ສະເໜີກ່ຽວກັບ Thread

- ຈຸດປະສົງທົ່ວໄປຂອງບັນດາພາສາຂຽນໂປຣແກຣມທັງຫຼາຍເຊັ່ນ: Java, C#, Visual C++ .NET, Visual Basic .NET ແລະ Python ແມ່ນເປັນເຄື່ອງພື້ນຖານໃນການສ້າງລະບົບເຮັດວຽກແບບຄູ່ຂະໜານໃຫ້ແກ່ນັກຂຽນໂປຣແກຣມ
- Multithreading
  - ນັກຂຽນໂປຣແກຣມສາມາດກຳໜົດສ້າງໂປຣແກຣມໃຫ້ມີຫຼາຍໆ threads ໄດ້
  - ແຕ່ລະ thread ຈະເປັນພາກສ່ວນໜຶ່ງຂອງໂປຣແກຣມທີ່ສາມາດເຮັດວຽກໄປພ້ອມໆກັບ threads ອື່ນ

ສະເໜີກ່ຽວກັບ Thread

- ລັກສະນທ່ົວໄປ
  - ແຕ່ເດີມນັ້ນ ຂະບວນການເຮັດໜ້າທີ່ຄວບຄຸມພຽງແຕ່ Thread ດຽວເທ່ົານັ້ນ ຈ່ຶງເອ້ີນວ່າ Heavyweight process
  - ຂະບວນການສາມາດຖ່ືກແຍກອອກເປັນຂະບວນການຍ່ອຍຫລາຍອັນຊ່ຶງເອ້ີນວ່າ Threads ບາງຄ້ັງເອ້ີນວ່າ Lightweight process
  - ການແຍກອອກເປັນຂະບວນການຍ່ອຍກ່ໍເພ່ືອໃຫ້ແຕ່ລະສ່ວນຍ່ອຍເຫລ່ົານັ້ນ ສາມາດເຮັດວຽກຕ່າງໄປພ້ອມໆກັນເພ່ືອໃຫ້ວຽກງານໄດ້ສໍາເລັດໄວຂ້ຶ້ນ
  - ແຕ່ລະ thread ຈະມ່ີ ໝາຍເລກ thread, ຕ່ືອນັບໂປຣແກຣມ, ກຸ່ມ Register, ບ່ອນເກ ບຂ້ໍມູນຊ່ົວຄາວເປັນຂອງຕ່ືວເອງ, ສ່ວນ address space ແລະ ບັນດາຂ້ໍມູນຮ່ວມກັນອ່ືນແມ່ນໃຊ້ຮ່ວມກັບພ່ໍແມ່ຂອງມັນ
  - Thread ~~ຈະເຮັດວຽກຮ່ວມກັບ thread~~ ອ່ືນໆໃນຂະບວນການດຽວ ກັນເຊ່ັນ: ລະຫັດ, ຂ້ໍມູນ ແລະ ຊັບພະຍາກອນອ່ືນໆ

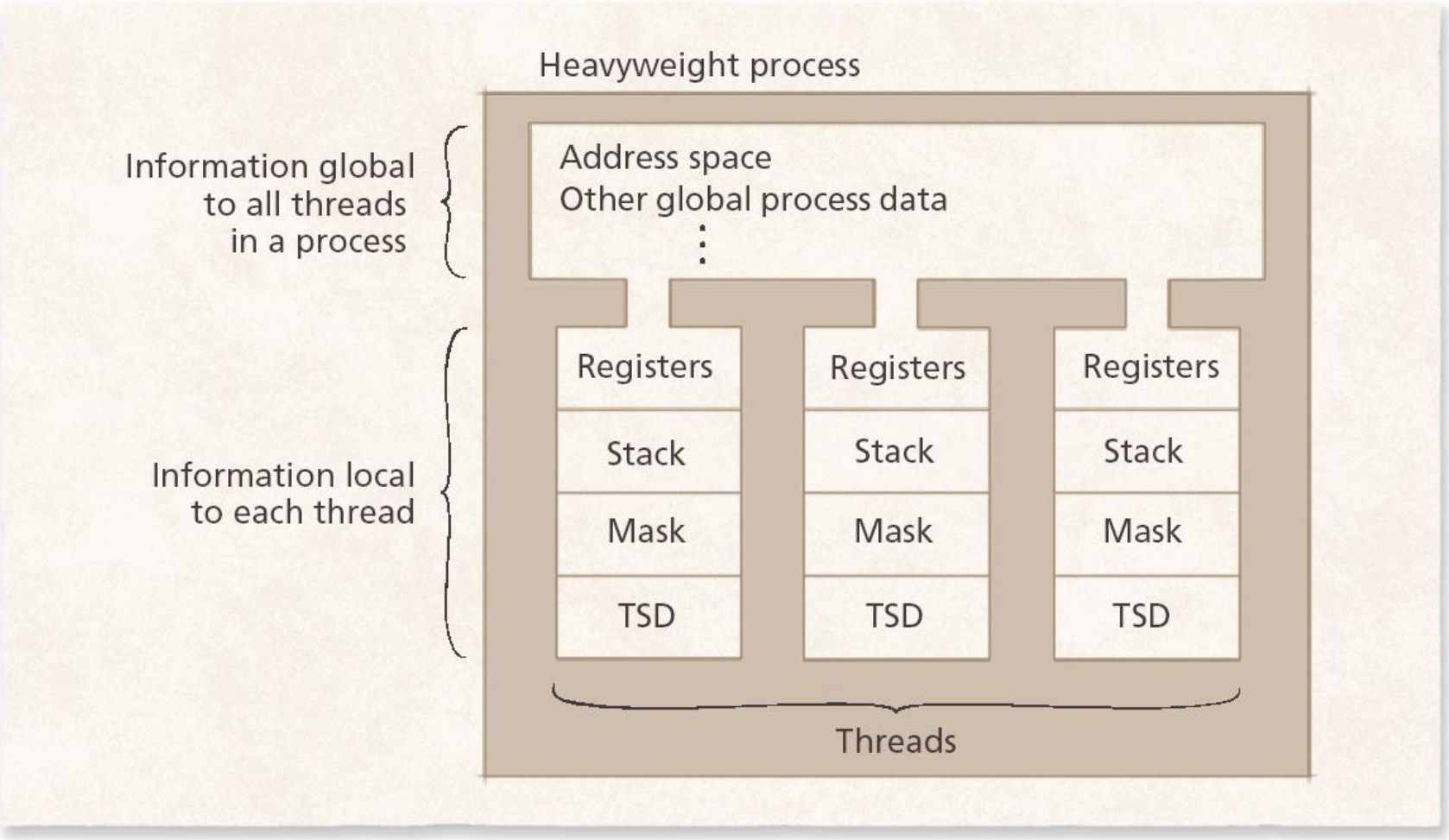
ສະເໜີກ່ຽວກັບ Thread

- ປະໂຫຍດ
  - ເຮັດໃຫ້ເວລາໃນການປະມວນຜົນໂດຍລວມໄວຂຶ້ນ ຊຶ່ງສົ່ງຜົນໃຫ້ເວລາໃນການຕອບສະໜອງຕໍ່ຜູ້ໃຊ້ໄວຂຶ້ນ
  - ໃຊ້ຊັບພະຍາກອນຮ່ວມກັນ
    - Thread ຕ່າງໆຈະໃຊ້ຊັບພະຍາກອນຮ່ວມກັບ thread ອື່ນໆໃນຂະບວນການດຽວກັນ, ມີການເອີ້ນໃຊ້ລະຫັດດຽວກັນ ດັ່ງນັ້ນ ກິດຈະກຳຕ່າງໆຂອງ thread ຈຶ່ງເກີດຂຶ້ນພາຍໃນຕຳແໜ່ງທີ່ຢູ່ດຽວກັນ
  - ປະຢັດໜ່ວຍຄວາມຈຳ ແລະ ຊັບພະຍາກອນ
    - Thread ທີ່ຖືກສ້າງຂຶ້ນມາຈະໃຊ້ຊັບພະຍາກອນ ແລະ ໜ່ວຍຄວາມຈຳຮ່ວມກັບ ພໍ່ແມ່ຂອງມັນ ເຮັດໃຫ້ປະຢັດກ່ວາ
  - ສາມາດໃຊ້ປະໂຫຍດຈາກສະຖາປັດຕະຍະກຳ Multiprocessor
    - ການເຮັດວຽກແບບ multithread ຈະຊ່ວຍເພີ່ມປະສິດທິພາບທີ່ຖືກອອກແບບມາໃນສະຖາປັດຕະຍະກຳແບບ multiprocessor ເພາະວ່າ thread ສາມາດເຮັດວຽກແບບພ້ອມໆກັນໄດ້

ນິຍາມຂອງ Thread

- Thread
  - ເປັນຂະບວນການຍ່ອຍ (Lightweight process (LWP))
  - ອາດຈະເປັນ thread ເພື່ອເຮັດວຽກໃດໜຶ່ງ ຫຼື thread ເພື່ອຄວບຄຸມ
  - ໃຊ້ເນື້ອທີ່ໜ່ວຍຄວາມຈຳ ແລະ ຂໍ້ມູນສ່ວນລວມ ຮ່ວມກັບຂະບວນການທີ່ສ້າງມັນຂຶ້ນມາ
  - ສ່ວນ Registers, stack, signal masks ແລະ ຂໍ້ມູນສະເພາະອື່ນເປັນຂອງໃຜລາວ
- ບັນດາ Thread ອາດຈະຖືກບໍລິຫານຈັດການໂດຍລະບົບປະຕິບັດການ ຫຼື ຜູ້ໃຊ້ໂປຣແກຣມ
- ຕົວຢ່າງ: Win32 threads, C-threads, Pthreads

ນິຍາມຂອງ Thread



ແຮງບັນດານໃຈໃນການສ້າງ Thread

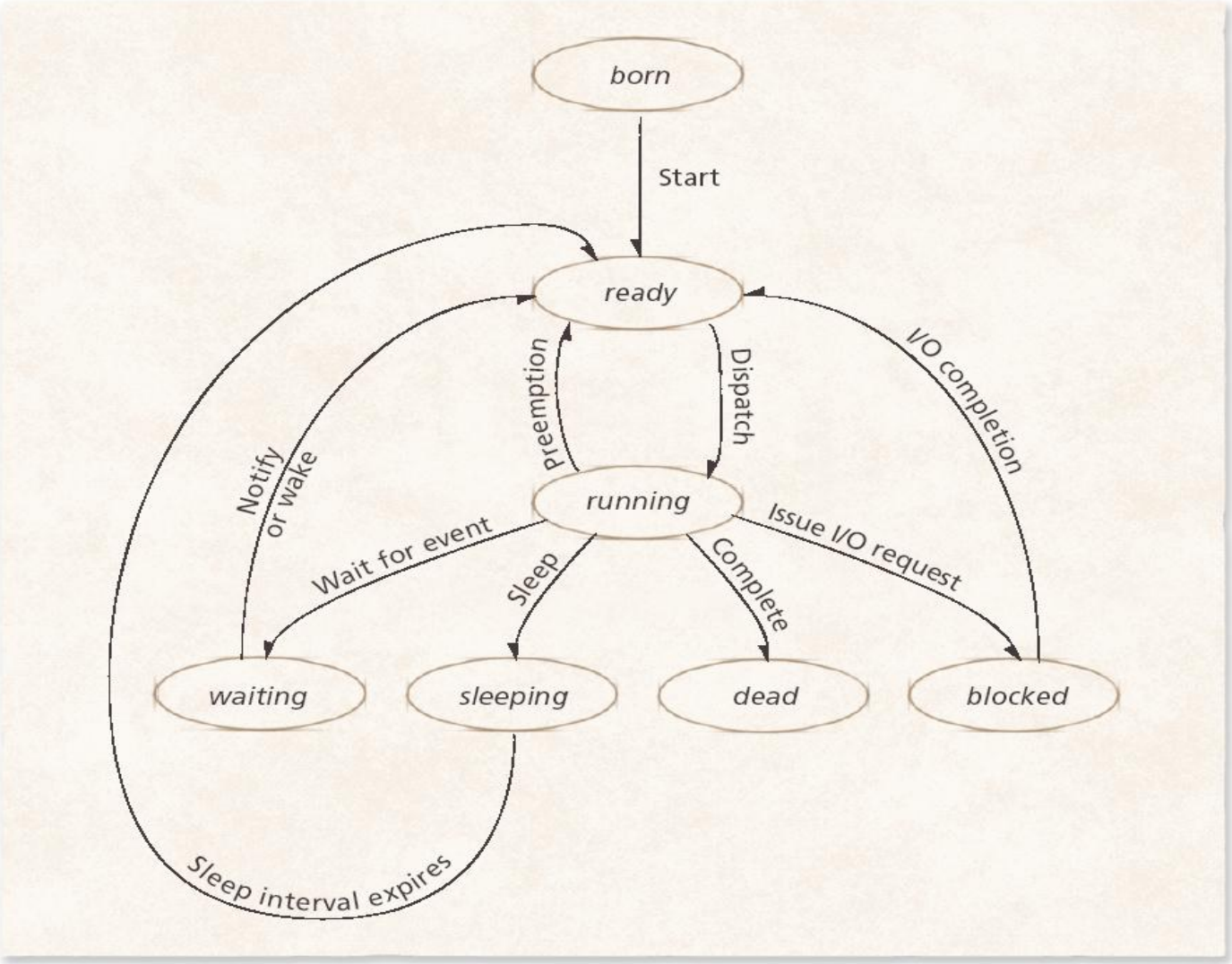
- Threads ໄດ້ຖືກໃຊ້ຫຼາຍຂຶ້ນເນື່ອງຈາກ
  - ການອອກແບບຊອບແວຣ໌ (Software design) ຈະເປັນແບບເຮັດໃຫ້ສາມາດເຮັດໄດ້ຫຼາຍວຽກເປັນຄູ່ຂະກັນ (parallel tasks)
  - ປະສິດທິພາບ (Performance) ໃນການເຮັດວຽກດີຂຶ້ນເນື່ອງຈາກສາມາດເອົາໄປໃຊ້ກັບລະບົບຫຼາຍໜ່ວຍປະມວນຜົນ (multiprocessor)
  - ສາມາດໃຊ້ຊັບພະຍາກອນບາງຢ່າງຮ່ວມກັນໄດ້ (Cooperation)
    - ໃຊ້ເນື້ອທີ່ໜ່ວຍຄວາມຈຳ ແລະ ຂໍ້ມູນບາງຢ່າງຮ່ວມກັນເຮັດໃຫ້ບໍ່ສິ້ນເປືອງ
  - ແຕ່ລະ thread ຈະເຮັດວຽກໄປຕາມແຕ່ລະສະຖານະພາບ
  - Threads ແລະ processes ມີຕົວດຳເນີນການຄືກັນ
  - ການສ້າງ Thread ລະບົບປະຕິບັດການບໍ່ຈຳເປັນຈັດສັນຊັບພະຍາກອນ
  - ບໍ່ສິ້ນເປືອງເມື່ອສົມທຽບກັບການສ້າງຂະບວນການໃໝ່

ສະຖານະພາບຂອງ Thread -ວົງຈອນຊີວິດ

- ສະຖານະພາບຂອງ Thread
  - Born state
  - Ready state (runnable state)
  - Running state
  - Dead state
  - Blocked state
  - Waiting state
  - Sleeping state

ສະຖານະພາບຂອງ Thread -ວົງຈອນຊີວິດ

- ສ



ການດຳເນີນການກ່ຽວກັບ Thread

- Threads ແລະ processes ມີຕົວດຳເນີນການຄືກັນ
  - Create
  - Exit (terminate)
  - Suspend
  - Resume
  - Sleep
  - Wake

ການດຳເນີນການກ່ຽວກັບ Thread

- ຕົວດຳເນີນການຂອງ Threads ທີ່ບໍ່ມີໃນ processes
  - Cancel
    - ເປັນການແຈ້ງເຕືອນວ່າ thread ນັນຄວນຈະຢຸດການເຮັດວຽກ ແຕ່ບໍ່ຮັບປະກັນວ່າມັນຢຸດການເຮັດວຽກ
    - Threads ສາມາດບໍ່ປະຕິບັດຕາມຄຳສັ່ງດັ່ງກ່າວ
  - Join
    - Thread ຫຼັກສາມາດລໍຖ້າບັນດາ threads ໃຫ້ສິ້ນສຸດການເຮັດວຽກດ້ວຍການ joining ກັບພວກມັນ
    - Thread ທີ່ Join ຈະລໍຖ້າຈົນກ່ວາ thread ທີ່ມັນເຂົ້າຮ່ວມສິ້ນສຸດການເຮັດວຽກ

Threading Model

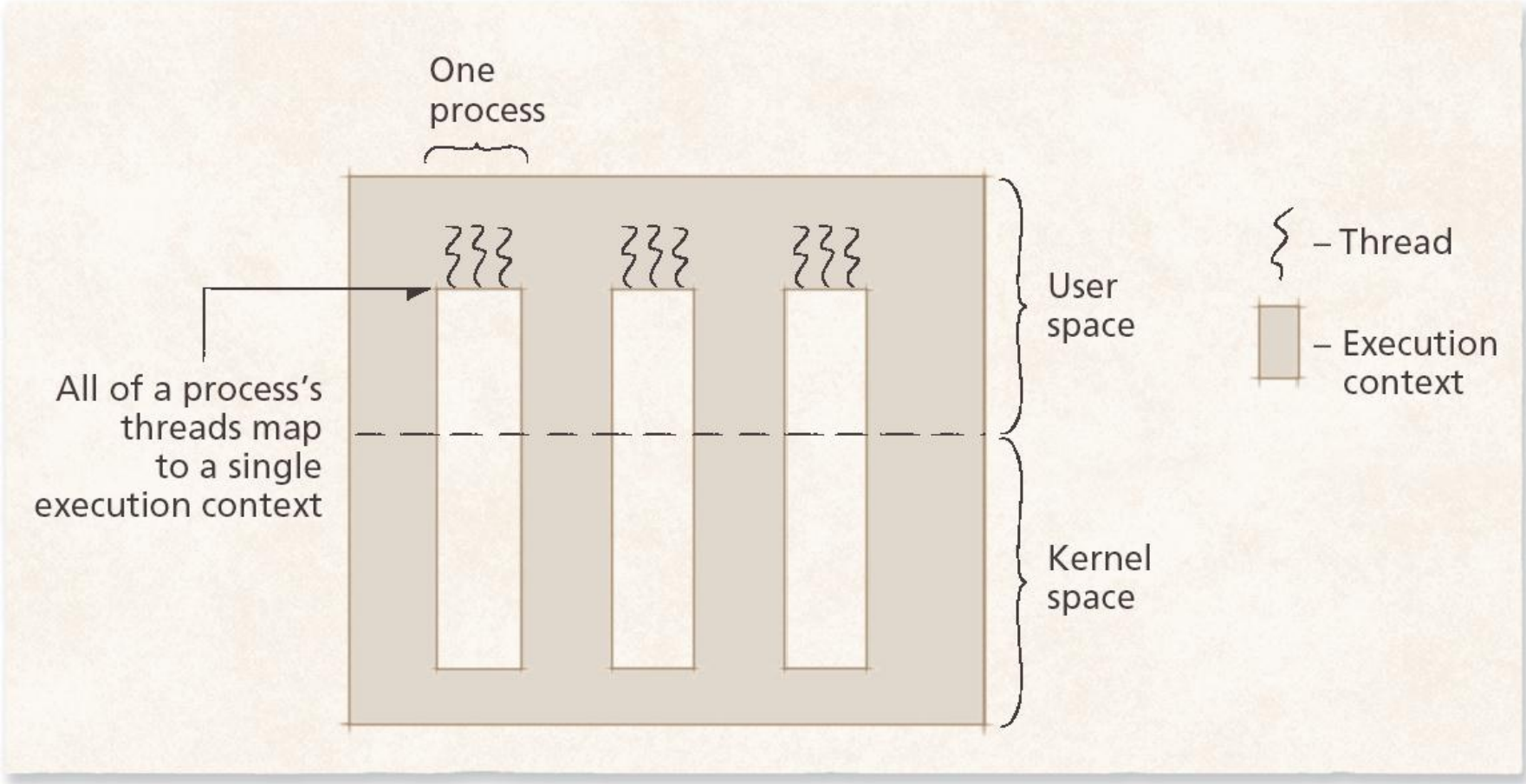
- ມີ 3 threading models ທີ່ໄດ້ຮັບຄວາມນິຍົມ
  - User-level threads
  - Kernel-level threads
  - Combination of user- and kernel-level threads
- User-level threads
  - User-level threads ເຮັດວຽກຢູ່ໃນ user space
    - Threads ຖືກສ້າງໂດຍ runtime libraries ທີ່ບໍ່ສາມາດໃຊ້ຄຳສັ່ງພິເສດ ຫຼື ເຂົ້າໄປໃຊ້ kernel ໄດ້ໂດຍກົງ

Threading Model

- User-level threads
  - ການສ້າງ User-level thread ເປັນແບບ
    - Many-to-one thread mappings
      - ລະບົບປະຕິບັດການຈະເຊື່ອມຕໍ່ທຸກ thread ໃນຂະບວນການ multithreaded ໄປຫາການປະມວນຜົນໃດໜຶ່ງ
      - Advantages
        - User-level libraries ສາມາດກຳໜົດຕາຕະລາເວລາໃຫ້ thread ຂອງມັນເພື່ອເຮັດປະສິດທິພາບດີທີ່ສຸດ
        - ການສັບປ່ຽນການໃຊ້ງານຊັບພະຍາກອນຮ່ວມກັນຈະເຮັດຢູ່ນອກ kernel, ບໍ່ໃຊ້ຫຼັກການ context switches ຊຶ່ງເຮັດໃຫ້ CPU ມີເວລາຫວ່າງໄປເຮັດວຽກອື່ນ
        - ສາມາດໃຊ້ໄດ້ກັບທຸກລະບົບປະຕິບັດການ
      - Disadvantage
        - Kernel ຈະເບິ່ງ multithreaded process ເປັນລັກສະນະ thread ຄວບຄຸມອັນໜຶ່ງ ຊຶ່ງສາມາດເຮັດໃຫ້ປະສິດທິພາບບໍ່ດີໃນລະບົບ ຫຼາຍໜ່ວຍປະມວນຜົນ ແລະ ບໍ່ສາມາດໃຊ້ໄດ້ກັບລະບົບຫຼາຍໜ່ວຍປະມວນຜົນ

Threading Model

- User-level threads

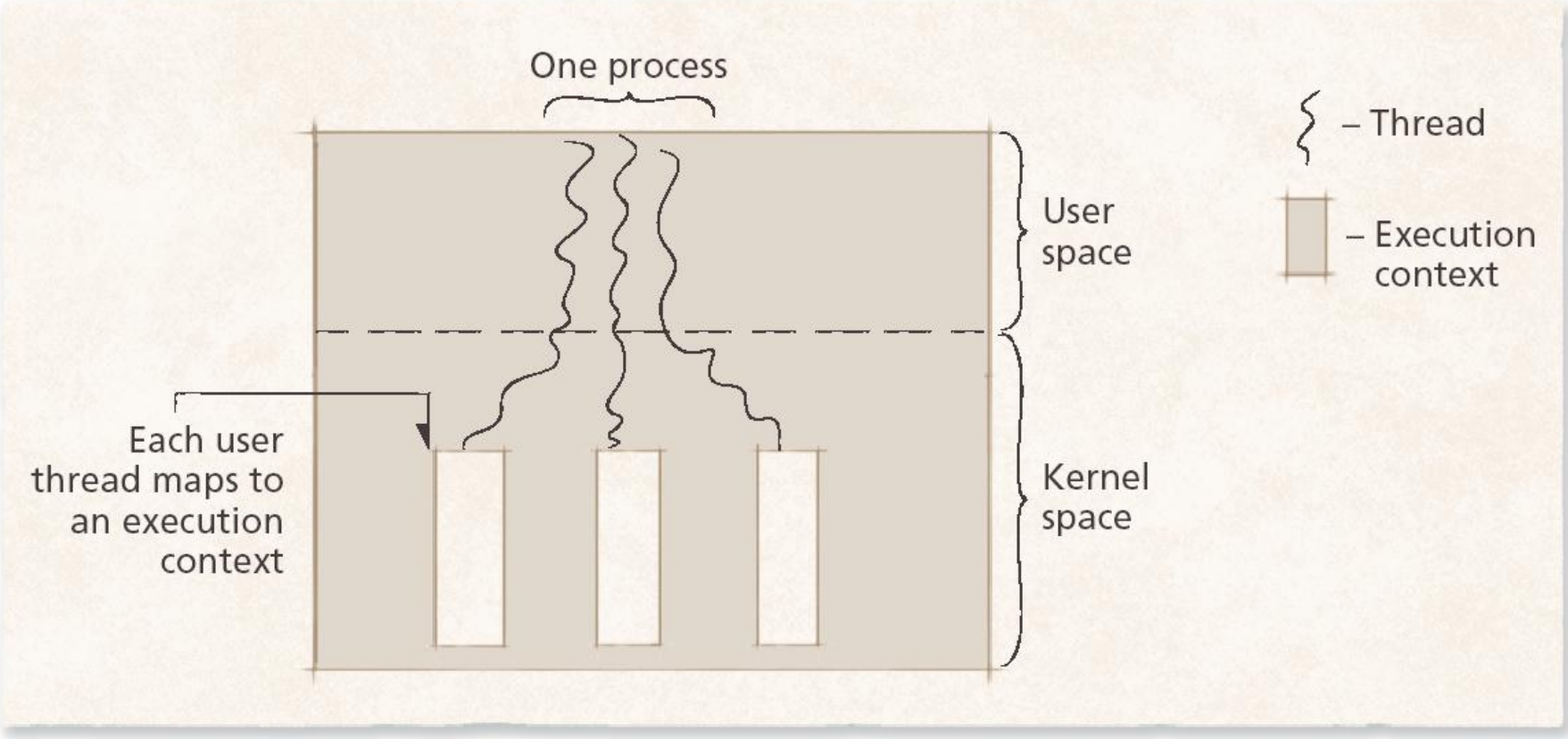


Threading Model

- Kernel-level threads
  - ເພື່ອແກ້ໄຂບັນຫາທີ່ມີໃນ user-level threads ໂດຍການເຊື່ອມຕໍ່ແຕ່ລະ thread ໄປຫາການປະມວນຜົນອັນໜຶ່ງ
    - Kernel-level threads ຈະເປັນແບບ one-to-one thread mapping
      - Advantages: ໃຊ້ໄດ້ກັບລະບົບຫຼາຍໜ່ວຍປະມວນຜົນI, ສາມາດຕອບສະໜອງຕໍ່ຜູ້ໃຊ້ໄດ້ຫຼາຍຂຶ້ນ, ປະລິມານການປະມວນຜົນຕໍ່ຫົວໜ່ວຍເວລາໄດ້ຫຼາຍຂຶ້ນ
      - Disadvantages: ສິ້ນເປືອງໃນການໃຊ້ງານ context switching ແລະ ຂຶ້ນກັບ APIs ຂອງລະບົບປະຕິບັດການ
  - Kernel-level threads ຈະບໍ່ແມ່ນທາງເລືອກທີ່ດີທີ່ສຸດສຳຫຼັບບັນດາໂປຣແກຣມຕ່າງໆ

Threading Model

- Kernel-level threads

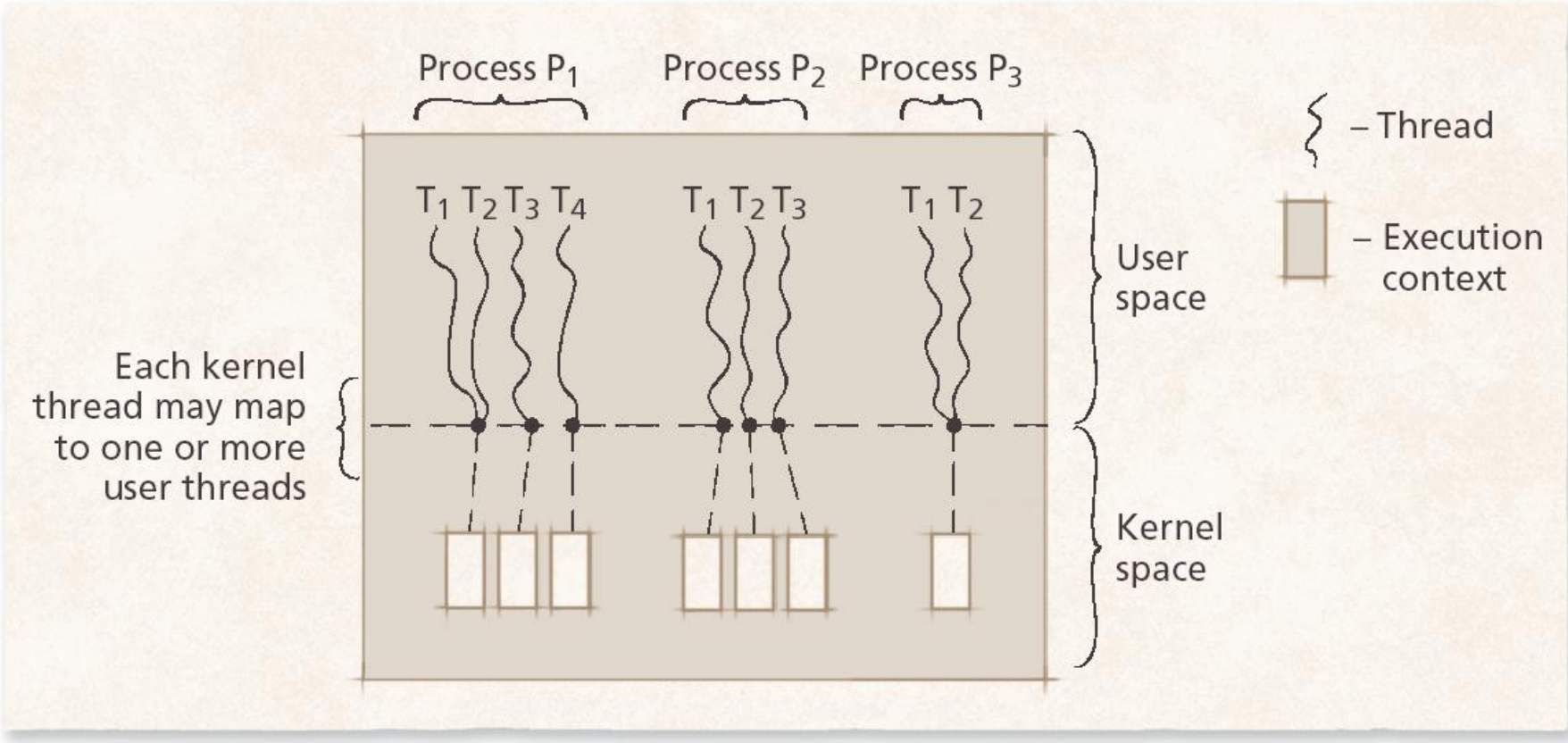


Threading Model

- ການປະສົມປະສານລະຫວ່າງ user- ແລະ kernel-level threads
  - ເປັນການເອົາສອງຮູບແບບມາປະສົມປະສານກັນຊຶ່ງເປັນແບບ Many-to-many thread mapping
    - ຈຳນວນຂອ user ແລະ kernel threads ບໍ່ຈຳເປັນຕ້ອງເທົ່າກັນ
    - ບໍ່ສິ້ນເປືອງເມື່ອທຽບກັບແບບ one-to-one ໂດຍການສ້າງ thread pooling
  - Worker threads
    - kernel threads ຈະອາໃສຢູ່ໃນ thread pool ໄດ້ຍາວນານ
    - ເຮັດໃຫ້ປະສິດທິພາບດີຂຶ້ນໃນສະພາບທີ່ມີການສ້າງແລະທຳລາຍເປັນປະຈຳ
    - ແຕ່ລະ thread ໃໝ່ໄດ້ຖືກສັ່ງໃຫ້ເຮັດວຽກດ້ວຍ worker thread
  - Scheduler activation
    - ເປັນເທັກນິກທີ່ເຮັດໃຫ້ user-level library ກຳໜົດຕາຕະລາງເວລາໃຫ້ແກ່ threads ຂອມັນ
    - ເຮັດວຽກເມື່ອລະບົບປະຕິບັດການເອີ້ນ user-level threading library ເພື່ອກຳໜົດວ່າ threads ຂອງມັນຕ້ອງການຈັດຕາຕະລາງຄືນໃໝ່ບໍ

Threading Model

- ການປະສົມປະສານລະຫວ່າງ user- ແລະ kernel-level threads

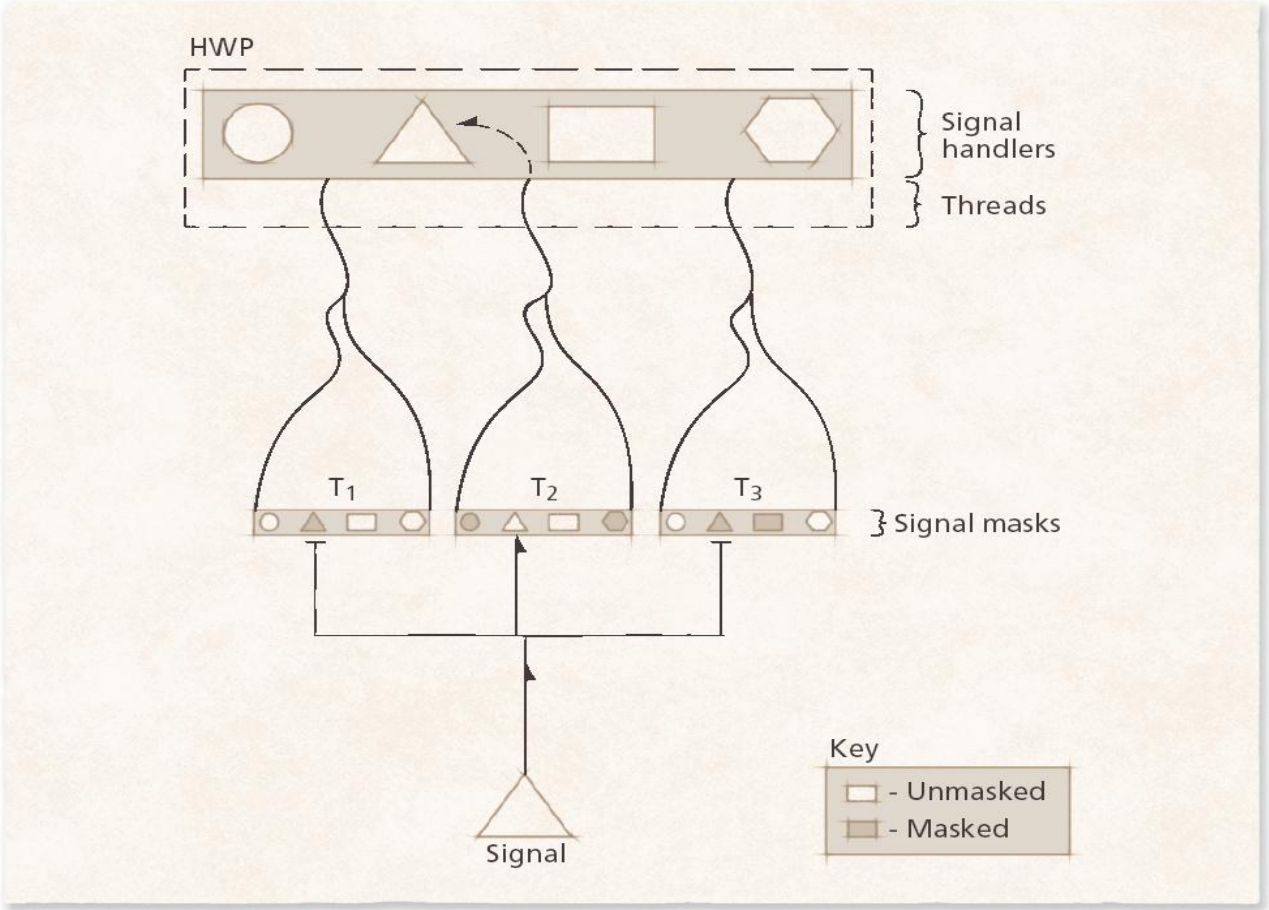


ສິ່ງທີ່ຄວນພິຈາລະນາໃນການສ້າງ Thread

- ການສົ່ງສັນຍານໄປຫາ thread
  - ມີສອງປະເພດສັນຍານ
    - Synchronous:
      - ເປັນສັນຍານສົ່ງໄປຫາ thread ທີ່ກ່ຽວພັນກັບການປະຕິບັດງານຄຳສັ່ງຂອງມັນໂດຍກົງ
      - ຈະຖືກສົ່ງຫາ thread ທີ່ກຳລັງປະຕິບັດງານຢູ່ໃນປະຈຸບັນ
    - Asynchronous
      - ເປັນສັນຍານສົ່ງໄປຫາ thread ທີ່ບໍ່ໄດ້ພົວພັນກັບການປະຕິບັດງານຄຳສັ່ງມັນ
      - Threading library ຈະຕ້ອງກຳໜົດວ່າແມ່ນ thread ຈະເປັນຜູ້ຮັບສັນຍານດັ່ງກ່າວ ເພື່ອໃຫ້ການສົ່ງຖືກຕ້ອງ
  - ແຕ່ລະ thread ຈະສຳພັນກັບບັນດາສັນຍານຈຳນວນໜຶ່ງທີ່ຈະຖືກສົ່ງຫາ ເມື່ອມັນປະຕິບັດງານ
  - Thread ຈະເລືອກທີ່ຈະບໍ່ຮັບສັນຍານສົ່ງມາຫາມັນ ຍົກເວັ້ນແຕ່ສັນຍານ ທີ່ມັນຕ້ອງການ

ສິ່ງທີ່ຄວນພິຈາລະນາໃນການສ້າງ Thread

- Thread Signal Delivery



ສິ່ງທີ່ຄວນພິຈາລະນາໃນການສ້າງ Thread

- ການສິ້ນສຸດການເຮັດວຽກຂອງ Thread (cancellation)
  - ກົງກັນຂ້າມກັບການສ້າງ Thread
  - ເກີດຂຶ້ນເມື່ອເຮັດວຽກສຳເລັດ
  - ອາດຈະສິ້ນສຸດກ່ອນກຳໜົດເນື່ອງຈາກໄປອ້າງອີງໜ່ວຍຄວາມຈຳຜິດບ່ອນ ຫຼື ສັ່ງໃຫ້ສິ້ນສຸດໂດຍຂະບວນການອື່ນ
  - ການສ້າງ thread ບາງອັນເຮັດໃຫ້ thread ກຳໜົດໄດ້ວ່າເມື່ອໃດມັນຈະຖືກເຮັດໃຫ້ສິ້ນສຸດ ເພື່ອປ້ອງກັນຂະບວນການບໍ່ໃຫ້ຢູ່ຜິດສະຖານະພາບ
  - ສະນັ້ນ threading library ຈະຕ້ອງຮູ້ຈັກວ່າຈະເອົາ thread ອອກຈາກລະບົບເມື່ອໃດ ແລະ ເຮັດແນວໃດ

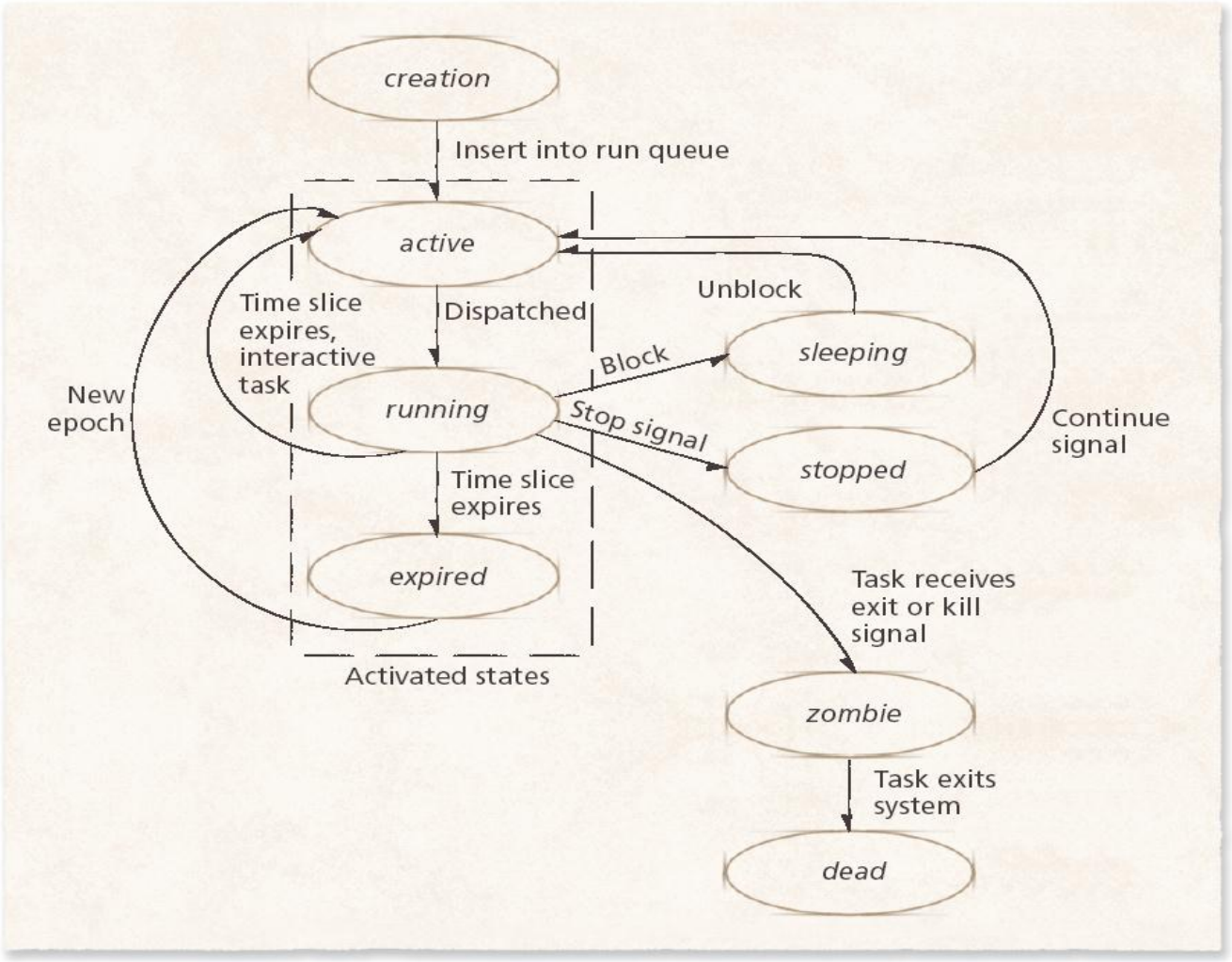
POSIX ແລະ Pthreads

- POSIX (Portable Operating System Interface for Computing Environment) ແມ່ນຍັນດາມາດຕະຖານສຳຫຼັບພາກສ່ວນສື່ສານຂອງລະບົບປະຕິບັດການ
- Threads ທີ່ໃຊ້ API ຂອງ POSIX threading ເອີ້ນວ່າ Pthreads
  - POSIX ໄດ້ຖືກກຳໜົດວ່າ processor registers, stack ແລະ signal mask ເປັນຂອງສະເພາະແຕ່ລະ thread
  - POSIX ໄດ້ຖືກກຳໜົດວ່ທີ່ລະບົບປະຕິບັດການຈະສົ່ງສັນຍານໄປຫາ Pthreads ເພີ່ມຕື່ມເພື່ອບອກໃຫ້ສິ້ນສຸດການປະຕິບັດງານແມ່ນຂຶ້ນກັບຮູບແບບໃນການສິ້ນສຸດການປະຕິບັດງານ

Linux Threads

- Linux allocates the same type of process descriptor to processes and threads (tasks)
- Linux uses the UNIX-based system call fork to spawn child tasks
- To enable threading, Linux provides a modified version named clone
  - Clone accepts arguments that specify which resources to share with the child task

Linux Threads



Windows XP Threads

- Threads
  - Actual unit of execution dispatched to a processor
  - Execute a piece of the process's code in the process's context, using the process's resources
  - Execution context contains
    - Runtime stack
    - State of the machine's registers
    - Several attributes

Windows XP Threads

- Windows XP threads can create fibers
  - Fiber is scheduled for execution by the thread that creates it, rather than the scheduler
- Windows XP provides each process with a thread pool that consists of a number of worker threads, which are kernel threads that execute functions specified by user threads

Windows XP Threads