ບົດທີ 6

ການຈັດຕາຕະລາງການເຮັດວຽກຂອງໜ່ວຍປະເມີນຜົນກາງ

(Processor Scheduling)

ເນື້ອໃນຫຍໍ້

- ສະເໜີເບື້ອງຕົ້ນ
- ລະດັບຂອງຕາຕະລາງເວລາ
- ຕາຕະລາງແບບ Preemptives ແລະ Nonpreemptive
- ລຳດັບຄວາມສຳຄັນຂອງຂະບວນການ (Priorities)
- ເປົ້າໝາຍຂອງການຈັດຕາຕະລາງ
- ເງື່ອນໄຂໃນການສ້າງຕາຕະລາງເວລາ
- Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ
- ຕາຕະລາງເວລາແບບມີກຳໜົດ
- ຕາຕະລາງເວລາແບບ Real-time

ສະເໜີເບື້ອງຕົ້ນ

- ຍຸດທະສາດໃນການຈັດຕາຕະລາງການໃຊ້ງານໜ່ວຍປະມວນຜົນ ຈະເປັນຜູ້ຕັດສິນວ່າຂະບວນການໃດຈະເຮັດວຽກໃນເວລາໃດ
- ຕາຕະລາງແຕ່ລະປະເພດຈະມີເປົ້າໝາຍຕ່າງກັນເຊັ່ນ:
  - ໃຫ້ປະມວນຜົນໄດ້ຫຼາຍທີ່ສຸດໃນຫົວໜ່ວຍເວລາໃດໜຶ່ງ
  - ໃຊ້ເວລາໃນການປະມວນຜົນໃຫ້ໜ້ອຍທີ່ສຸດ
  - ປ້ອງກັນບໍ່ໃຫ້ມີການເລື່ອນການເຮັດວຽກໄປຕະຫຼອດການ
  - ກຳໜົດເວລາການເຮັດວຽກໃຫ້ແຕ່ລະຂະບວນການ
  - ເຮັດໃຫ້ການໃຊ້ໜ່ວຍປະມວນຜົນມີປະສິດທິພາບສຸງສຸດ

ລະດັບຂອງຕາຕະລາງເວລາ

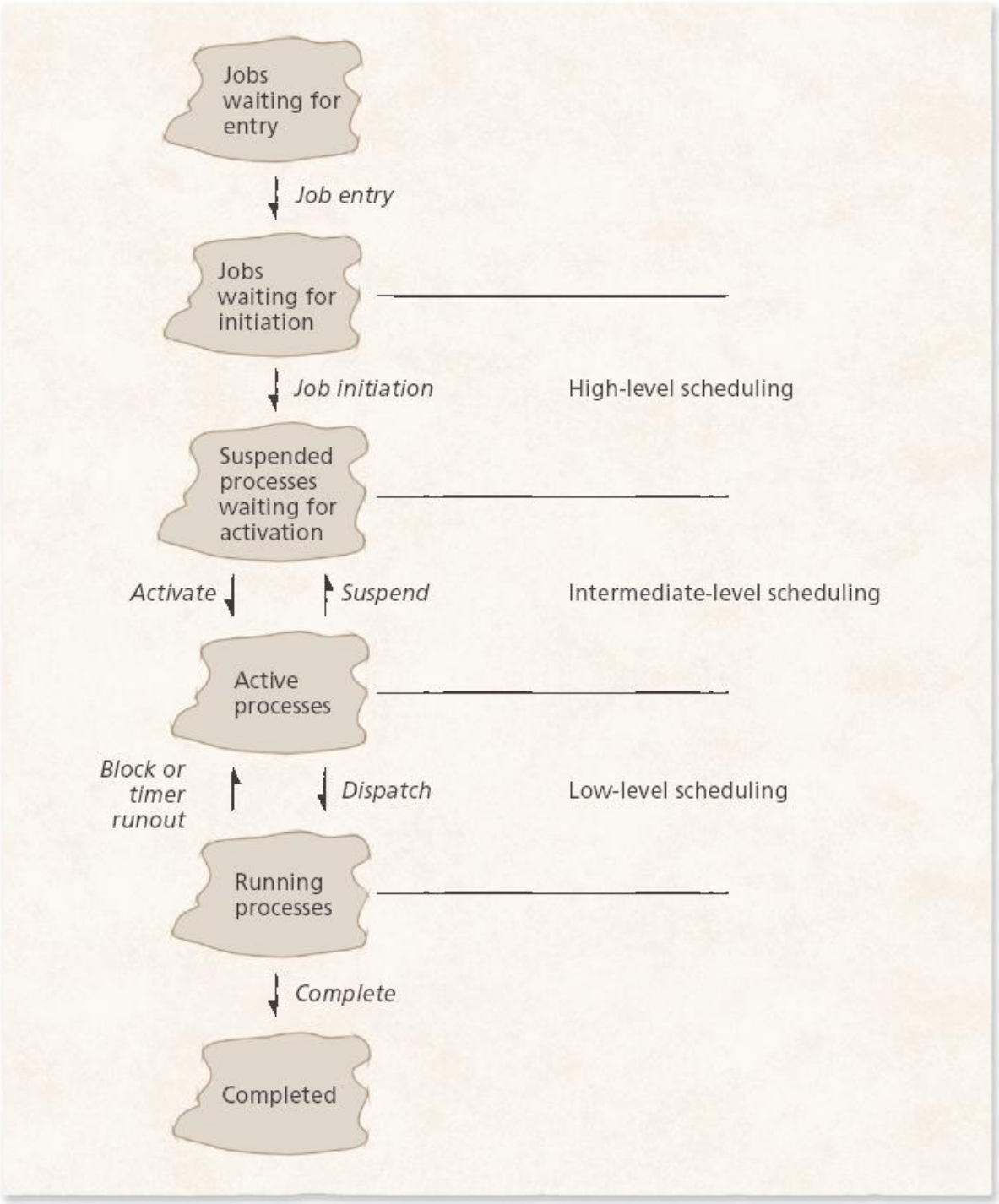
- High-level scheduling
  - ຈະເປັນຜູ້ກຳໜົດວ່າຂະບວນການໃດຈະຮັບອະນຸຍາດໃຫ້ເຂົ້າມາໃນລະບົບ
  - ຈະຖືກໃຊ້ເມື່ອຂະບວນການໃໝ່ຖືກສ້າງຂຶ້ນມາ
  - ໃຊ້ຄວບຄຸມຈຳນວນຂອງຂະບວນການທີ່ອະນຸຍາດໃຫ້ເຮັດວຽກພ້ອມກັນໃນລະບົບໃນເວລາໃດໜຶ່ງ
- Intermediate-level scheduling
  - ໃຊ້ເພື່ອເລືອກວ່າຂະບວນການໃດຄວນຈະຖືກອະນຸຍາດໃຫ້ເຂົ້າໄປລ່ຽນຢູ່ໃນຄິວກຽມພ້ອມ
  - ຖືກໃຊ້ເປັນປະຈຳ ຊຶ່ງໃຊ້ໃນການສັບປ່ຽນຂະບວນການເຂົ້າອອກ
  - ເປັນພາກສ່ວນໜຶ່ງຂອງ swapping function (ໃຊ້ເປັນ buffer)

ລະດັບຂອງຕາຕະລາງເວລາ

- Low-level scheduling
  - ໃຊ້ເພື່ອກຳໜົດວ່າຂະບວນການໃດທີ່ກຳລັງລໍຖ້າຢູ່ໃນຄິວກຽມພ້ອມ ຈະໄດ້ເຂົ້າໄປໃຊ້ CPU ເມື່ອມັນຫວ່າງ
  - ຖືກເອີ້ນໃຊ້ເມື່ອ
    - ມີສັນຍານໂມງຂັດຈັງຫວะ (Clock Interrupt)
    - ມີສັນຍານຂັດຈັງຫວะຈາກ I/O (I/O Interrupt)
    - ການເອີ້ນໃຊ້ຈາກລະບົບປະຕິບັດການ (Operating System Calls)
    - ສັນຍານຈາກຂະບວນການອື່ນ

ລະດັບ

- Low-le


Jobs waiting for entry
Job entry
Jobs waiting for initiation
Job initiation
High-level scheduling
Suspended processes waiting for activation
Activate
Suspend
Intermediate-level scheduling
Active processes
Block or timer runout
Dispatch
Low-level scheduling
Running processes
Complete
Completed

ตาตะลางแบบ Preemptives ແລະ Nonpreemptive

- Preemptive processes
  - ສາມາດເອົາຂະບວນການດັ່ງກ່າວອອກຈາກໜ່ວຍປະມວນຜົນຖ້າເຫັນວ່າຈະຜົນປະໂຫຍດທາງດ້ານເວລາ
  - ເຮັດໃຫ້ເວລາຕອບສະໜອງດີຂຶ້ນ
  - ເໝາະສົມກັບລະບົບໂຕະຕອບໂດຍກົງ (interactive environments)
  - ຂະບວນການທີ່ເອົາອອກມາຈະຍັງເກັບໄວ້ໃນໜ່ວຍຄວາມຈຳຫຼັກ
- Nonpreemptive processes
  - ຂະບວນການດັ່ງກ່າວຈະຕ້ອງເຮັດວຽກຈົນສຳເລັດຈຶ່ງຈະອອກຈາກໜ່ວຍຄວາມຈຳ ຫຼື ໝົດເວລາໃນການໃຊ້ໜ່ວຍປະມວນຜົນ
  - ເຮັດໃຫ້ຂະບວນການທີ່ບໍ່ສຳຄັນກັນຂະບວນການທີ່ສຳຄັນຕະຫຼອດໄປ

ລຳດັບຄວາມສຳຄັນຂອງຂະບວນການ

- ລຳດັບຄວາມສຳຄັນແບບຄົງທີ່ (Static priorities)
  - ເປັນລຳດັບຄວາມສຳຄັນທີ່ກຳໜົດໃຫ້ຂະບວນການທີ່ບໍ່ປ່ຽນແປງ
  - ສ້າງງ່າຍ
  - ບໍ່ສິ້ນເປືອງ
  - ບໍ່ປ່ຽນຕາມເຫດການ ແລະ ສະພາບແວດລ້ອມ
- ລຳດັບຄວາມສຳຄັນແບບປ່ຽນແປງ (Dynamic priorities)
  - ເປັນລຳດັບຄວາມສຳຄັນທີ່ປ່ຽນຕາມເຫດການ ແລະ ສະພາບແວດລ້ອມ
  - ປ່ຽນແປງຕາມຄວາມເໝາະສົມ
  - ສິ້ນເປືອງກ່ວາການສ້າງລຳດັບຄວາມສຳຄັນແບບຕາຍຕົວ
  - ປັບປ່ຽນໂດຍການເພີ່ມຄ່າລຳດັບຄວາມຄັນຂະບວນການ

ເປົ້າໝາຍຂອງການຈັດຕາຕະລາງ

- ມີຫລາຍເປົ້າໝາຍຂຶ້ນຢູ່ກັບລະບົບ
  - ເພື່ອໃຫ້ເຮັດວຽກໄດ້ຫລາຍທີ່ສຸດໃນຫົວໜ່ວຍເວລາໃດໜຶ່ງ
  - ເພື່ອໃຫ້ຈຳນວນການສື່ສານລະຫວ່າງຂະບວນການທີ່ໄດ້ຮັບອັດຕາການຕອບສະໜອງທີ່ຍອມຮັບໄດ້ຫລາຍທີ່ສຸດ
  - ເພື່ອໃຫ້ໃຊ້ຊັບພະຍາກອນໃຫ້ເກີດປະໂຫຍດສຸງສຸດ
  - ເພື່ອບໍ່ໃຫ້ມີການເລື່ອນການເຮັດວຽກຂະບວນການໃດໜຶ່ງໄປຕະຫລອດການ
  - ເພື່ອໃຫ້ຂະບວນການທີ່ມີລຳດັບຄວາມສຳຄັນສຸງກ່ວາໄດ້ປະມວນຜົນກ່ອນ
  - ເພື່ອໃຫ້ສິ້ນເປືອງໜ້ອຍທີ່ສຸດ
  - ເພື່ອໃຫ້ສາມາດຄາດການລ່ວງໜ້າໄດ້

ເປົ້າໝາຍຂອງການຈັດຕາຕະລາງ

- ມີຫລາຍເປົ້າໝາຍປົກກະຕິທົ່ວໄປຂອງທຸກ Scheduler
  - ຄວາມຍຸດຕິທຳ (Fairness)
    - ທຸກໆຂະບວນການທີ່ຄ້າຍຄືກັນຈະໄດ້ຮັບການຈັດການເຊັ່ນດຽວກັນ ແລະ ບໍ່ໃຫ້ມີການເລື່ອນການຖືກປະຕິບັດໄປຕະຫລອດການ
  - ຄວາມສາມາດໃນການຄາດການລ່ວງໜ້າໄດ້ (Precictability)
    - ແຕ່ລະຂະບວນການຄວນໄດ້ເຂົ້າເຮັດວຽກໃນຊ່ວງເວລາເທົ່າໆກັນໃນແຕ່ລະເທື່ອ Load ໝາຍເຖິງວ່າສາມາດຄາດຄະເນໄດ້ລ່ວງໜ້າວ່າມັນຈະແລ້ວໃນເມື່ອໃດ
  - ຄວາມສາມາດໃນການຂະຫຍາຍລະບົບ (Scalability)
    - ມີຄວາມສາມາດໃນການຮອງຮັບການເຂົ້າມາໃຊ້ຂອງຜູ້ໃຊ້ຈຳນວນຫລາຍໆ

ເງື່ອນໄຂໃນການສ້າງຕາຕະລາງເວລາ

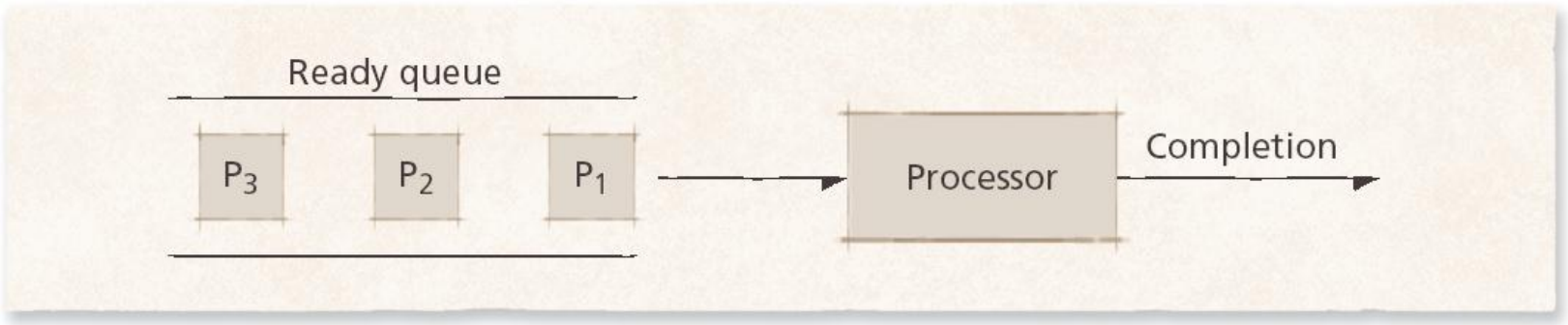
- ເປັນຂໍ້ກຳໜົດໃນການເລືອກໃຊ້ນະໂຍບາຍແບບໃດໜຶ່ງໃນການຈັດຕາຕະລາງເວລາເພື່ອໃຫ້ໃຊ້ງານ CPU ເຕັມປະສິດທິພາບຕາມສະຖານະການຕ່າງໆ
- ເງື່ອນໄຂການຕັດສິນໃຈໃນການຈັດຕາຕະລາງໄລຍະສັ້ນໄດ້ແບ່ງອອກເປັນ 2 ປະເພດ
  1. ເບິ່ງຕາມຜູ້ໃຊ້ເປັນຫລັກ (user-oriented)
     - ແມ່ນການກຳໜົດໃຫ້ພຶດຕິກຳຂອງລະບົບໃຫ້ເປັນທີ່ຍອມຮັບໄດ້ຂອງຜູ້ໃຊ້ທຸກຄົນແລະ ທຸກ ປະເພດ ເຊັ່ນ: ອັດຕາການຕອບສະໜອງ, ເວລາທີ່ໃຊ້ທັງໝົດ, ແລ້ວຕາມກຳໜົດເວລາ , ສາມາດຄາດການໄດ້
  2. ເບິ່ງຕາມລະບົບເປັນຫລັກ (system-oriented)
     - ແມ່ນການກຳໜົດໃຫ້ພຶດຕິກຳຂອງລະບົບໃຫ້ໃຊ້ CPU ເຕັມປະສິດທິພາບ ເຊັ່ນ: ການເຮັດວຽກໄດ້ຫຼາຍສຸດໃນຫົວໜ່ວຍເວລາໃດໜຶ່ງ, ອັດຕາມການເຮັດວຽກຂອງ CPU, ຄວາມ ຍຸດຕິທຳ, ການເບິ່ງຕາມຄວາມສຳຄັນ, ການແບ່ງປັນຊັບພະຍາກອນໃຫ້ດຸ່ນດ່ຽງ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- Algorithm ໃນການຈັດຕາຕະລາງ
  - ເປັນຜູ້ຕັດສິນວ່າແຕ່ລະຂະບວນການຈະເຮັດວຽກເມື່ອໃດ ແລະ ດົນປານໃດ
  - ເປັນຜູ້ເລືອກກ່ຽວກັບ
    - ຈະເອົາຂະບວນການອອກມາຈາກ CPU (Preemptibility)
    - ລຳດັບຄວາມສຳຄັນ (Priority)
    - ເວລາເຮັດວຽກ (Running time)
    - ການເຮັດວຽກຈົນສຳເລັດ (Run-time-to-completion)
    - ຄວາມຍຸດຕິທຳ (fairness)

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜມາກ່ອນໄດ້ເຮັດກ່ອນ (First-Come, First-Served: FCFS)
  - ເປັນຂັ້ນຕອນວິທີທີ່ງ່າຍ ແລະ ກົງໄປກົງມາ
  - ຂະບວນການໃດມາຮອດກ່ອນຈະໄດ້ຮັບ CPU ກ່ອນ
  - ເປັນແບບບໍ່ບັງຄັບ
  - ບໍ່ຄ່ອຍຖືກໃຊ້ເປັນວິທີຫລັກ



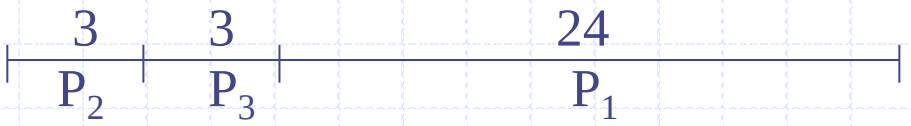
Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜມາກ່ອນໄດ້ເຮັດກ່ອນ (First-Come, First-Served: FCFS)
  - ຕົ້ວຢ່າງ: ມີຂະບວນດານທີ່ມາເຖິງຄືວລໍຖ້າພ້ອມກັນໃນເວລາ 0 ຕາມລຳດັບດັ່ງນີ້: P1, P2, P3 . ໂດຍມີເວລາໃນການເຂົ້າໃຊ້ CPU ມີຫົວໜ່ວຍເປັນວິນາທີດັ່ງນີ້: P1 = 24, P2 = 3, P3 = 3. ຈົ່ງຄຳນວນຫາ ເວລ່ລໍຖ້າ, ເວລາລໍຖ້າສະເລ່ຍ, ເວລາທັງໝົດ, ປະລິມານວຽກໃນຫົວໜ່ວຍເວລາ
  - ຕອບ:
    - ໄດ້ເສັ້ນເວລາດັ່ງນີ້

    - ເວລາລໍຖ້າ: $P_1 = 0$ $P_2 = 24$ $P_3 = 27$
    - ເວລາລໍຖ້າສະເລ່ຍ: $(0+24+27)/3 = 17$
    - ເວລາທັງໝົດ: $(24+27+30)/3 = 27$
    - ປະລິມານວຽກໃນຫົວໜ່ວຍເວລາ: $30/3 = 10$

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜມາກ່ອນໄດ້ເຮັດກ່ອນ (First-Come, First-Served: FCFS)
  - ຖ້າລຳດັບການເຂົ້າໃຊ້ CPU ແມ່ນ: P2, P3, P1 . ຈົ່ງຄຳນວນຫາປັດໃຈຕ່າງດັ່ງກ່າວ
  - ຕອບ:
    - ໄດ້ເສັ້ນເວລາດັ່ງນີ້



    - ເວລາລໍຖ້າ: $P_1 = 6 \quad P_2 = 0 \quad P_3 = 3$
    - ເວລາລໍຖ້າສະເລ່ຍ: $(6+0+3) / 3 = 3$
    - ເວລາໂດຍສະເລ່ຍ: $(3 + 6 + 30) / 3 = 13$
    - ປະລິມານວຽກໃນໜ່ວຍເວລາ: $30 / 3 = 10$

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜໃຊ້ເວລາໜ້ອຍໄດ້ເຮັດກ່ອນ (Shortest-Job-First: SJF)
  - ສາມາດລຸດເວລາໃນການລໍຖ້າລົງຖ້າທຽບກັບ FCFS
  - ໃຊ້ໄດ້ດີໃນກໍລະນີຫລາຍມາຮອດພ້ອມໆກັນ
  - ຖ້າມີ 2 ຂະບວນການທີ່ໃຊ້ເວລາເທົ່າກັນກໍ່ຈະໃຊ້ວິທີ FCFS ເຂົ້າມາຊ່ວຍ
  - ເປັນໄດ້ທັງແບບ ບໍ່ບັງຄັບ ແລະ ແບບບັງຄັບ
  - ບັນຫາກໍ່ຄື ຂະບວນການທີ່ເຮັດວຽກຫລາຍຈະບໍ່ມີໂອກາດໄດ້ເຂົ້າປະມວນຜົນ
  - ບໍ່ເໝາະສົມທີ່ຈະໃຊ້ງານກັບລະບົບ Interactive ລຸ້ນໃໝ່

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜໃຊ້ເວລາໜ້ອຍໄດ້ເຮັດກ່ອນ (Shortest-Job-First: SJF)
  - ຕົ້ວຢ່າງ: ກໍລະນີທີ 1

ຂະບວນການ:	P1	P2	P3	P4
ເວລາມາຮອດ:	0	2	4	5
ເວລາໃຊ້ CPU:	7	4	1	4

  - ຈັດຕາຕະລາງສຳຫລັບ non-preemptive SJF:

  - ເວລາໃນການລໍຖ້າ: 0 3 6 7
  - ເວລາລໍຖ້າສະເລ່ຍ: $(0 + 3 + 6 + 7)/4 = 4$
  - ເວລາທັງໝົດ: $0 + 7$ $6 + 4$ $3 + 1$ $7 + 4$
  - ເວລາໂດຍສະເລ່ຍ: $(7 + 4 + 10 + 11)/4 = 8$

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜໃຊ້ເວລາໜ້ອຍໄດ້ເຮັດກ່ອນ (Shortest-Job-First: SJF)
  - ຕົ້ວຢ່າງ: ກໍລະນີທີ 1

ຂະບວນການ :	P1	P2	P3	P4
ເວລາມາຮອດ:	0	2	4	5
ເວລາໃຊ້ CPU :	7	4	1	4

  - ຈັດຕາຕະລາງສຳຫລັບ preemptive SJF:

P_1	P_2	P_3	P_2	P_4	P_1

  - ເວລາໃນການລໍຖ້າ : 9 1 0 2
  - ເວລາລໍຖ້າສະເລ່ຍ : $(9 + 1 + 0 + 2)/4 = 3$
  - ເວລາທັງໝົດ : $9 + 7$ $1 + 4$ $0 + 1$ $2 + 4$
  - ເວລາໂດຍສະເລ່ຍ : $(16 + 5 + 1 + 6)/4 = 7$

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜໃຊ້ເວລາໜ້ອຍໄດ້ເຮັດກ່ອນ (Shortest-Job-First: SJF)
  - ວຽກບ້ານ

ໃຫ້ຈັດຕາຕະລາງໃຫ້ແກ່ຂະບວນການເຫລົ່ານີ້ ທັງແບບບັງຄັບ ແລະ ບໍ່ບັງຄັບ

ຂະບວນການ :	P1	P2	P3	P4	P5
ເວລາມາຮອດ:	0	3	4	7	8
ເວລາໃຊ້ CPU :	8	4	2	6	4

ຄຳນວນຫາເວລາລໍຖ້າ, ເວລາລໍຖ້າສະເລ່ຍ ແລະ ເວລາທັງໝົດຂອງແຕ່ລະຂະບວນການ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜສຳຄັນກ່ວາໄດ້ເຮັດກ່ອນ (Priority Scheduling)
  - ເປັນການກຳໜົດໃຫ້ແຕ່ລະຂະບວນການມີໝາຍເລກຄວາມສຳຄັນ ໂດຍການກຳໜົດໃຫ້ຕົວເລກນ້ອຍມີຄວາມສຳຄັນຫຼາຍ, ຕົວເລກໃຫຍ່ມີຄວາມສຳຄັນໜ້ອຍ ຫຼື ກົງກັນຂ້າມກໍ່ໄດ້
  - ຂະບວນການທີ່ມີຄວາມສຳຄັນຫລາຍກ່ອນຈະໄດ້ຖືກປະມວນຜົນກ່ອນ, ຕົວທີ່ມີຄວາມສຳຄັນໜ້ອຍຈະຖືກປະມວນຜົນຕາມຫລັງ
  - ຖ້າມີຫລາຍຂະບວນການມີຄວາມສຳຄັນເທົ່າກັນກໍ່ຈະໃຊ້ວິທີ FCFS, SJF ມາຊ່ວຍ
  - ເປັນໄດ້ທັງແບບບັງຄັບ ແລະ ບໍ່ບັງຄັບ
  - ດີກ່ວາວິທີ SJF
  - ບັນຫາ: ເກີດການລໍຖ້າແບບບໍ່ມີກຳໜົດ ຊຶ່ງສາມາດແກ້ໄຂໄດ້ດ້ວຍການເພີ່ມລຳດັບຄວາມສຳຄັນໃຫ້ແກ່ຂະບວນການທີ່ມີຄວາມສຳຄັນໜ້ອຍ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຜສຳຄັນກ່ອາໄດ້ເຮັດກ່ອນ (Priority Scheduling)
  - ສົມມຸດວ່າມີ 5 ຂະບວນການ

    P1 (ມີເວລາໃຊ້ CPU = 10, ລຳດັບຄວາມສຳຄັນ = 3),

    P2 (ມີເວລາໃຊ້ CPU = 1, ລຳດັບຄວາມສຳຄັນ = 1),

    P3 (ມີເວລາໃຊ້ CPU = 2, ລຳດັບຄວາມສຳຄັນ = 3),

    P4 (ມີເວລາໃຊ້ CPU = 1, ລຳດັບຄວາມສຳຄັນ = 4),

    P5 (ມີເວລາໃຊ້ CPU = 5, ລຳດັບຄວາມສຳຄັນ = 2).

    ຈະຈັດຕາຕະລາງການເຂົ້າໃຊ້ CPU ແນວໃດ?

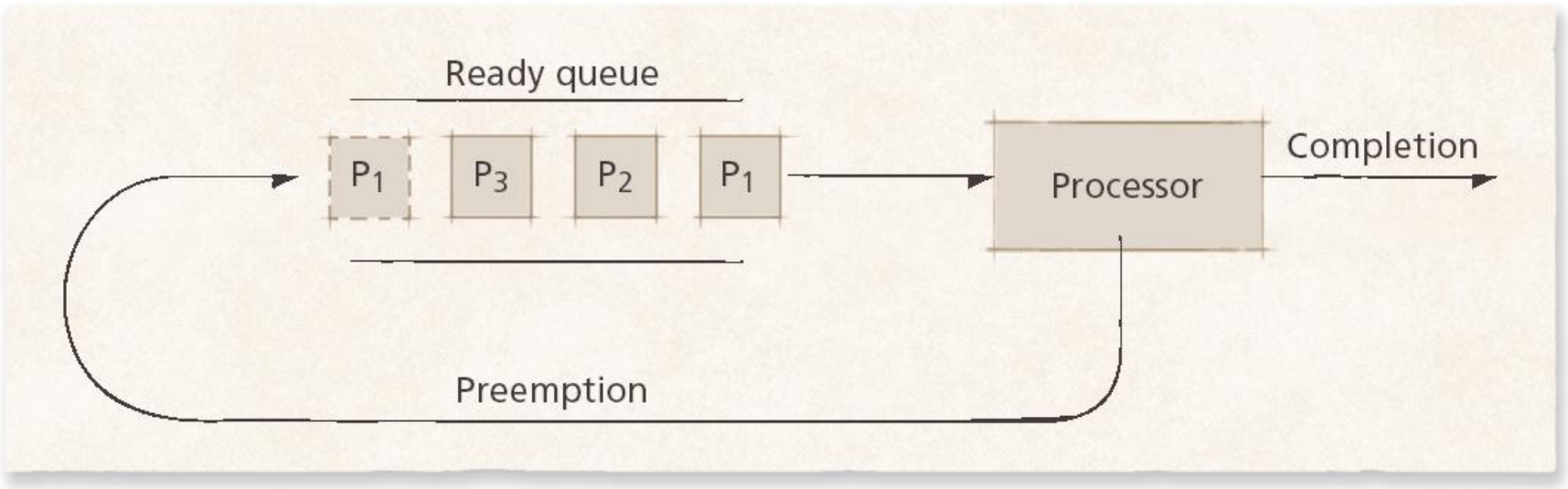
ເປັນວຽກບ້ານ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ປ່ຽນກັນເປັນຮອບວຽນ (Round-Robin Scheduling)
  - ອອກແບບມາສຳຫລັບລະບົບ Time Sharing
  - ເປັນແບບ FCFS ແຕ່ເພີ່ມວິທີໃຫ້ເປັນແບບບັງໃຫ້ສະຫລັບຂະບວນການໄດ້
  - ໄດ້ແບ່ງເວລາອອກເປັນສ່ວນໆເອີ້ນວ່າ Time Quantum ຫຼື Time Slice
  - ເມື່ອໝົດເວລາລະບົບຈະສະຫລັບຂະບວນການ ເອົາຂະບວນການທີ່ປະມວນຜົນຢູ່ໃນ CPU ໄປຕໍ່ຄິວຢູ່ຫລັງສຸດ ແລ້ວເອົາຂະບວນການທີ່ຢູ່ໜ້າຄິວເຂົ້າໄປປະມວນຜົນຕໍ່
  - ຖ້າມີ n ຂະບວນການຢູ່ໃນຄິວ ແລະ Time Slice = q, ເວລາໃນການລໍຖ້າຈະບໍ່ເກີນ (n-1)q
  - ປະສິດທິພາບຂອງວິທີນີ້ຂຶ້ນຢູ່ກັບຄ່າຂອງ q, ຖ້າ q ໃຫຍ່ຫາຍຈະຄືກັບ FCFS, ແຕ້ຖ້າ q ນ້ອຍຫຼາຍຈະເຮັດໃຫ້ເສຍເວລາໃນການສະຫລັບກັນ
  - ເວລາສະເລ່ຍຂອງແຕ່ລະຂະບວນການຈະສຸງກ່ວາວິທີ SJF, ແຕ່ຕອບສະໜອງຈະດີກ່ວາ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ປ່ຽນກັນເປັນຮອບວຽນ (Round-Robin Scheduling)



Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ປ່ຽນກັນເປັນຮອບວຽນ (Round-Robin Scheduling)
  - ຈົ່ງຄຳນວນຫາ ເວລາລໍຖ້າໃນວິທີຂອງ round robin ສຳຫຼັບ 3 ຂະບວນການ ຕໍ່ໄປນີ້

    P1 (ມີເວລາໃຊ້ CPU = 24),

    P2 (ມີເວລາໃຊ້ CPU = 3),

    P3 (ມີເວລາໃຊ້ CPU = 4)

    time quantum = 4.

  ແລະ ຊອກຫາເວລາລໍຖ້າໂດຍສະເລ່ຍ ແລະ ລອງປ່ຽນລຳດັບໃນຄິວເບິ່ງວ່າກໍລະນີ ໃດທີ່ເວລາລໍຖ້ານ້ອຍທີ່ສຸດ?

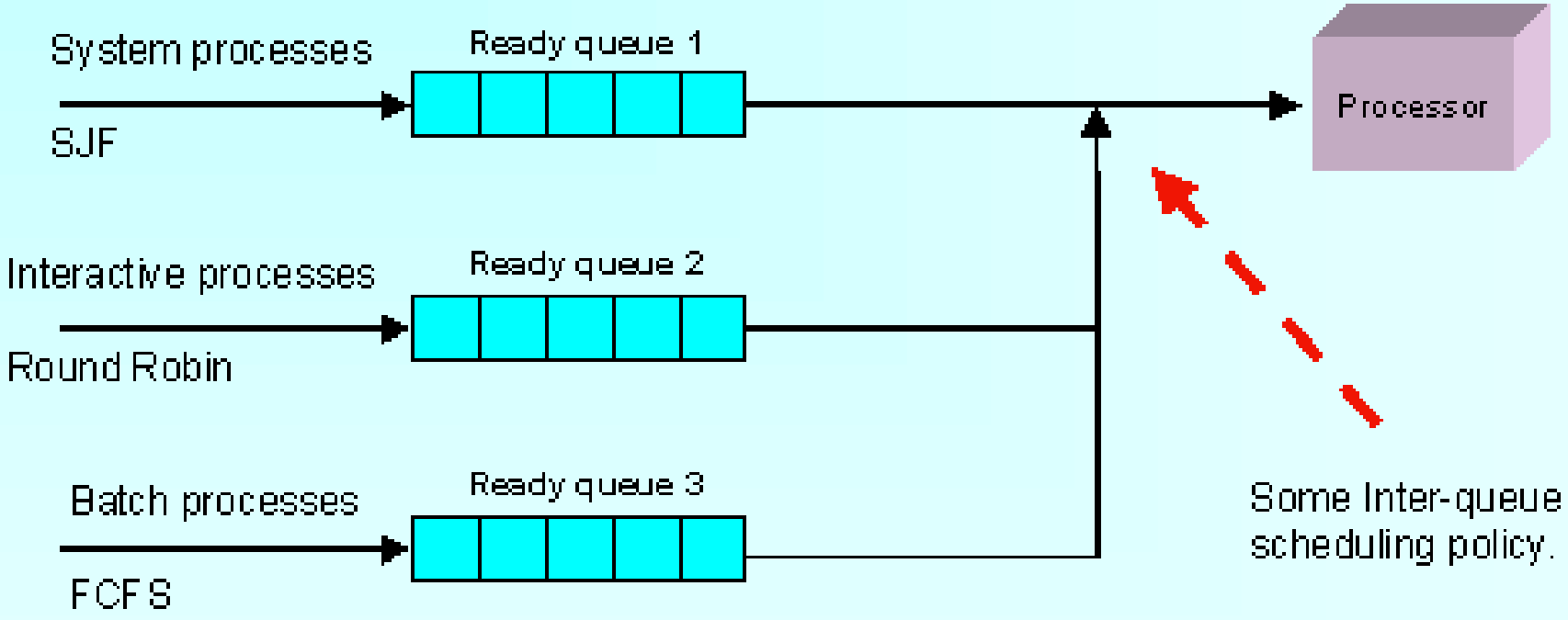
ເປັນວຽກບ້ານ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຊ້ຄິວຫລາຍລະດັບ (Multilevel Queue Scheduling)
  - ເປັນການແບ່ງຂະບວນການທັງໝົດອອກເປັນກຸ່ມ ແຍກຕາມລຳດັບຄວາມສຳຄັນຂອງວຽກ, ຂະໜາດຂອງໜ່ວຍຄວາມຈຳທີ່ໃຊ້, ຄວາມສຳຄັນຂອງຂະບວນການ, ປະເພດຂອງຂະບວນການ…
  - ເມື່ອຂະບວນການໃດຖືກຈັດຢູ່ໃນກຸ່ມໃດແລ້ວຈະຢູ່ໃນກຸ່ມນັ້ນຕະຫລອດໄປ ບໍ່ສາມາດປ່ຽນໄດ້
  - ແບ່ງຄິວລໍຖ້າອອກເປັນຫລາຍອັນສຳຫລັບຂະບວນການແຕ່ລະກຸ່ມ
  - ແຕ່ລະຄິວລໍຖ້າຈະມີວິທີຈັດລຳດັບເປັນຂອງຕົນເອງ ຫຼືອາດແຕກຕ່າງຈາກຄິວອື່ນ ຊຶ່ງຂຶ້ນຢູ່ກັບຂະບວນການວ່າເປັນແບບໃດ

Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຊ້ຄິວຫລາຍລະດັບ (Multilevel Queue Scheduling)

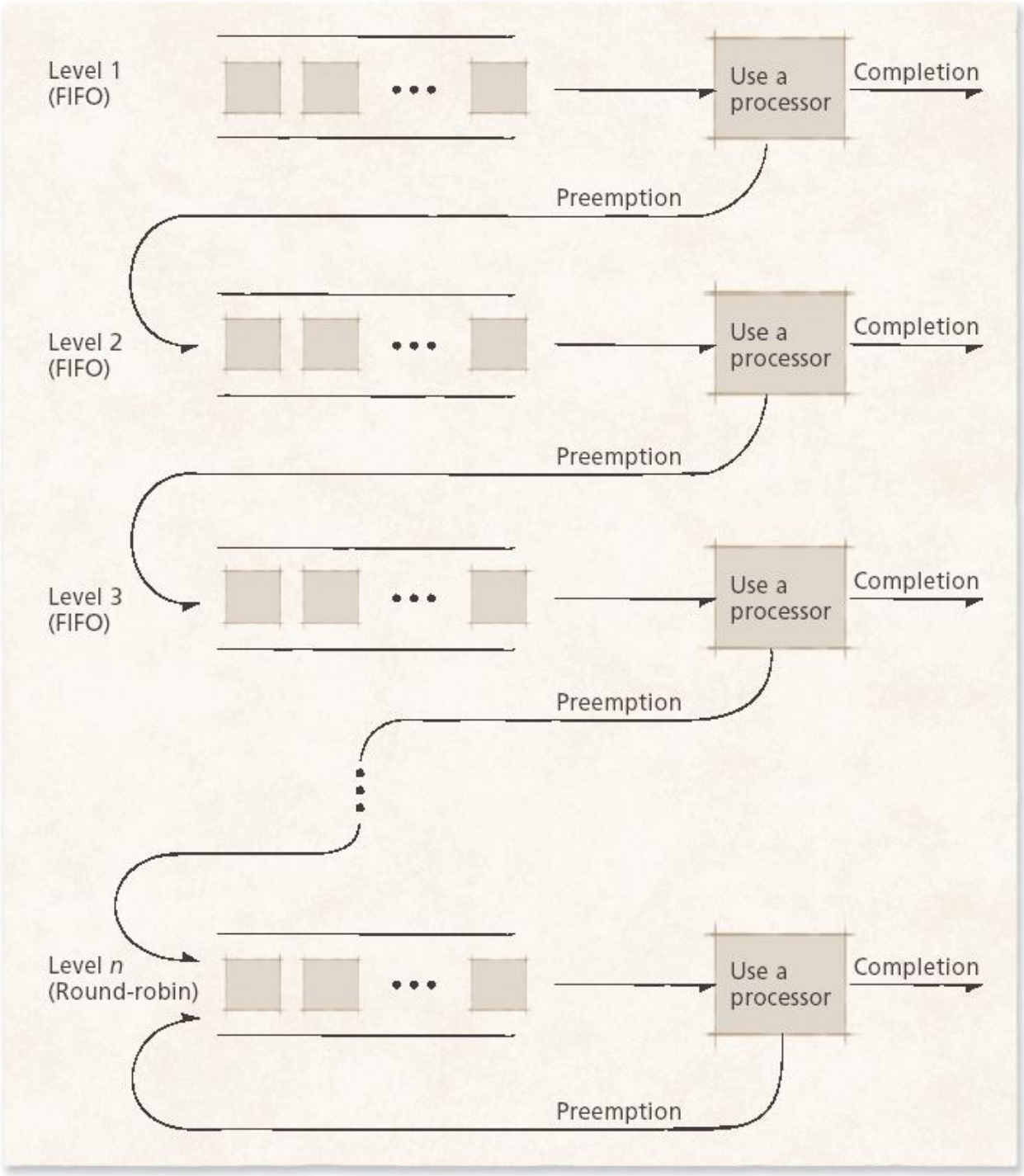


Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ໃຊ້ຄິວປ້ອນກັບຫລາຍລະດັບ (Multilevel Feedback-Queue Scheduling)
  - ຄືກັບຄິວແບບຫລາຍລະດັບແຕ່ສາມາດໃຫ້ມີການຍ້າຍຂ້າມຄິວໄດ້
  - ໃນແຕ່ລະຄິວແມ່ນໃຊ້ວິທີແບບ FCFS
  - ການແບ່ງກຸ່ມຂອງຂະບວນການໃຫ້ແຕກຕ່າງກັນຕາມຄຸນລັກສະ ນະຂອງການໃຊ້ເວລາໃນ CPU ເປັນສຳຄັນ. ຂະບວນການໃດທີ່ໃຊ້ເວລາໃນ CPU ຫຼາຍ ຈະຖືກຍ້າຍໄປຫາຄິວທີ່ມີລຳດັບຄວາມສຳຄັນຕ ່ຳລົງໄປ ແລະ ກົງກັນຂ້າມ

Algor ๆ

- ใช้ถือป้ eue
  Sched



Algorithm ໃນການຈັດຕາຕະລາງປະເພດຕ່າງໆ

- ຕາຕະລາງເວລາແບບມີກຳໜົດ(Deadline Scheduling)
  - ຂະບວນການຕ້ອງເຮັດວຽກສຳເລັດພາຍໃນເວລາທີ່ກຳໜົດ
  - ໃຊ້ເມື່ອຜົນການປະມວນຜົນຈະບໍ່ຖືກໃຊ້ຖ້າບໍ່ສົ່ງມອບພາຍໃນເວລາທີ່ກຳໜົດ
  - ສ້າງຍາກ
    - ຈະຕ້ອງໄດ້ວາງແຜນຊັບພະຍາກອນທີ່ຕ້ອງການໃຊ້ໄວ້ລ່ວງໜ້າ
    - ສິ້ນເປືອງຫຼາຍ
    - ປະສິດທິພາບໃນການໃຫ້ບໍລິການແກ່ຂະບວນການອື່ນໆຈະຕ່ຳລົງ

ການຈັດຕາຕະລາງເທິງລະບົບ Multiprocessor

- ໃຊ້ວິທີກະຈາຍວຽກໄປຫາທຸກໆ CPU ໂດຍການແບ່ງຄິວລໍຖ້າໄປໃຫ້ແຕ່ລະ ຕົວແບບເທົ່າທຽມກັນ
- ເພື່ອປ້ອງກັນບໍ່ໃຫ້ມີ CPU ຕົວໃດຕົວໜຶ່ງຫວ່າງຢູ່ ໃນຄະນະທີ່ຕົວອື່ນເຮັດ ວຽກໜັກ ຈະຕ້ອງໃຫ້ມີຄິວລໍຖ້າສ່ວນກາງ ເພື່ອຈັດສັນໃຫ້ CPU ທີ່ຫວ່າງ
- ສາມາດເລືອກໃຊ້ວິທີການຈັດຕາຕະລາງແບບ ການປະມວນຜົນເທົ່າທຽມກັນ , ຫຼື ແບບບໍ່ເທົ່າທຽມກັນ ກໍ່ໄດ້
  - ແບບເທົ່າທຽມກັນ, CPU ແຕ່ລະຕົວຈະມີວິທີການຈັດລຳດັບຂະບວນການເຂົ້າມາຄິວ ລໍຖ້າ ແລະ ການເລືອກເຂົ້າປະຕິບັດງານເປັນຂອງຕົນເອງ ແຕ່ຕ້ອງຮັບປະກັນວ່າບໍ່ໃຫ້ມີ CPU ຫຼາຍຕົວໄປເອົາຂະບວນການດຽວກັນ ແລະ ບໍ່ໃຫ້ມີຂະບວນການໃດເສຍໄປ ຈາກຄິວ
  - ແບບບໍ່ເທົ່າທຽມກັນ, ຈະຕ້ອງມີ CPU ຕົວໜຶ່ງເປັນຕົວຫລັກ ທີ່ເຮັດໜ້າທີ່ ຕັດສິນໃຈ ໃນການຈັດລຳດັບ, ການປະມວນຜົນ I/O ແລະ ຄວບຄຸມກິດຈະກຳອື່ນໆທັງໝົດໃນ ລະບົບ ສ່ວນຕົວອື່ນແມ່ນເຮັດຕາມ ຕົວຫລັກແບ່ງໃຫ້

ການຈັດຕາຕະລາງໃນລະບົບ Real Time

- ລະບົບ Real Time ແບບເຄັ່ງຄັດ
  - ເປັນການຮັບປະກັນວ່າວຽກທີ່ສຳຄັນທີ່ສຸດຈະຕ້ອງສຳເລັດພາຍໃນເວລາທີ່ກຳໜົດ
  - Scheduler ຈະສົ່ງວຽກດັ່ງກ່າວເຂົ້າໄປປະມວນຜົນຕໍ່ເມື່ອຮັບປະກັນວ່າວຽກດັ່ງກ່າວຈະແລ້ວພາຍໃນເວລາທີ່ກຳໜົດ ຖ້າບໍ່ດັ່ງນັ້ນຈະຕ້ອງປະຕິເສດໄປ
  - ລະບົບຈະຕ້ອງກມີການຈອງຊັບພະຍາກອນລ່ວງໜ້າ ໂດຍທີ Scheduler ກໍ່ຕ້ອງຮູ້ຢ່າງຊັດເຈັນວ່າ ຄຳຮ້ອງຂໍນັ້ນຈະໃຊ້ເວລາໜ້ອຍຫລາຍປານໃດໃນການເອີ້ນໃຊ້ Function ຕ່າງໆຂອງລະບົບປະປະຕິບັດການ ຫ ຼ ແຕ່ລະກິດຈະກຳໃຊ້ເວລາຫຼາຍສຸດບໍ່ເກີນເທົ່າໃດ, ກິດຈະກຳທີ ສາມາດເຮັດໃຫ້ສຳເລັດໄດ້

ການຈັດຕາຕະລາງໃນລະບົບ Real Time

- ລະບົບ Real Time ແບບບໍ່ເຄັ່ງຄັດ
  - ລຸບົບນີ້ຕ້ອງການພຽງແຕ່ໃຫ້ວຽກທີ່ສຳຄັນທີ່ສຸດມີລຳດັບຄວາມສຳຄັນທີ່ສຖງກ່ວາຂະບວນການທົ່ວໄປກ່ໍພໍ
  - Scheduler ຈະຕ້ອງໄດ້ຖືກອອກແບບຢ່ຳງລະມັດລະວັງ
    - ລະບົບຈະຕ້ອງເຮັດຕາຕະລາງແບບເບິ່ງລຳດັບຄວາມສຳຄັນ ແລະ ຂະບວນການເຮັດວຽກແບບ Real Time ທັງໝົດຈະຕ້ອງໄດ້ຮັບຄວາມສຳຄັນສຸງສຸດ
    - ເວລາລໍຖ້ຳປະມວນຜົນຈະຕ້ອງນ້ອຍທີ່ສຸດ
  - ຖ້ຳຕ້ອງການໃຫ້ເວລາລໍຖ້ຳນ້ອຍທີ່ສຸດ ຈະຕ້ອງໃຫ້ System calls ສາມາດຄວບຄຸມໄດ້ດ້ວຍວິທີໃດໜຶ່ງເຊັ່ນ:
    - ການໃສ່ຈຸດບັງຄັບ ເຂ້ຳໄປໃນ System calls ເພື່ອຂັດຈັງຫວະໃນຄະນະທີ່ມີຂະບວນການມີລຳດັບຄວາມສຳຄັນສຸງສຸດຕ້ອງການປະມວນຜົນ
    - ໃຫ້ kernel ທັງໝົດສາມາດຄວບຄຸມໄດ້ ໂດຍຈະປ້ອງກັນຂໍ້ມູນຂອງ kernel ຈາກຂະບວນການທີ່ສຳຄັນດ້ວຍຫລັກການປະສານເວລາ