№5 лаб сұрақтарға жауап

1. Шартты қабылдау (яғни жоғарыдағы «жоғарыдағы процестер» деген процестер)

Мен суреттен және тапсырмадан алынған ықтимал бастапқы процестерді келесідей қабылдадым (ең ықшам әрі анық мысал): • P1 — Arrival = 0, Burst = 5 • P2 — Arrival = 2, Burst = 3 • P3 — Arrival = 4, Burst = 1

(Егер сенің нақты басқа мәндерің болса — айтшы, солармен қайта есептеймін.)

1. Multilevel Queue (MLQ) тәсілі туралы қысқаша түсінік • MLQ — бірнеше деңгейлі кезек (мысалы, жоғары приоритетті system және төмен приоритетті user). • Әр деңгейдің өзінің жоспарлаушысы болуы мүмкін (мысалы, system үшін FCFS, user үшін Round-Robin). • Әдетте жоғары приоритетті кезек (system) төмен приоритетті кезектерге қарағанда басымдыққа ие: яғни жоғары приоритетті кезекте тапсырмалар бар болса, олар алдымен орындалады; төмен приоритеттегі процестер жоғары приоритет өзінің барынша орындалғаннан кейін ғана орындалады. (Бұл — классикалық не-преэмптив емес MLQ мінезі.)

Менің қабылдаған бөлімдеуім және әр-queue саясаттары • system (жоғары приоритет): P1 және P3 (мысал үшін). Политика — FCFS (non-preemptive). • user (төмен приоритет): P2. Политика — FCFS (қарапайым әрі түсінікті; егер керек — оны RR ретінде де көрсетеміз).

(Ескерту: процестерді system/user бағдарламашы өзі бөледі; мен мысал үшін осылай бөлдім. Қажет болса басқа бөлу жасап қайта есептеймін.)

1. Кестелік және уақыттық есеп (Gantt диаграммасы)

Біз жоғарыда айтқан non-preemptive MLQ ережесін қолданамыз: жоғары приоритетті кезек босамайынша төмен приоритеткі шақырылмайды.

Келісілген келу уақытына сай орындалу хронологиясы: 1. Уақыт 0: P1 (system) келеді және басталады. • P1 толығымен орындалып бітеді: 0 → 5 (burst = 5). 2. Уақыт 2-де P2 (user) келеді — user кезегіне орналастырылады (бірақ орындау кейінге қалады, себебі systemде P1 бар). 3. Уақыт 4-де P3 (system) келеді — system кезегіне қосылады. 4. P1 аяқталғаннан кейін (уақыт 5), system кезегінде P3 бар — сондықтан P3 орындалады. • P3: 5 → 6 (burst = 1). 5. Енді system кезегі босайды → төмен приоритетті user кезегі орындала алады. • P2: 6 → 9 (burst = 3).

Gantt chart (уақыт бойынша): 0 — 5 : P1 | 5 — 6 : P3 | 6 — 9 : P2

1. Әр процесс үшін нәтижелер (Finish, Turnaround, Waiting)

Есептеулер: • P1: • Arrival = 0, Finish = 5 • Turnaround = Finish − Arrival = 5 − 0 = 5 • Waiting = Turnaround − Burst = 5 − 5 = 0 • P2: • Arrival = 2, Finish = 9 • Turnaround = 9 − 2 = 7 • Waiting = 7 − 3 = 4 • P3: • Arrival = 4, Finish = 6 • Turnaround = 6 − 4 = 2 • Waiting = 2 − 1 = 1 12:00 Орташа мәндер: • Орташа Turnaround = (5 + 7 + 2) / 3 = 4.67 • Орташа Waiting = (0 + 4 + 1) / 3 = 1.67

(Шамалар ондыққа дейін дөңгелектенген.)

1. Түсініктеме: нәтижелер неліктен осылай? • P1 жүйелік (system) және бірінші келгендіктен дереу орындалып бітеді → күту = 0. • P3 де жүйелік, бірақ P1 аяқталғаннан кейін ғана орындалды, сондықтан оның TAT шағын (1 орындау + 1 уақыт күту = TAT 2). • P2 қолданбалы (user) болғандықтан ол жоғары приоритетті system кезектері бітпегенше орындауға кірмеді — сондықтан оның күту уақыты ең көп.
2. Егер әр queue басқа саясат қолданса (мысалдар) • Мысалы: system — FCFS, user — Round-Robin (quantum=2). Бұл жағдайда P2 орындау кезінде бөлу болуы мүмкін емес, себебі P2 бір ғана user процесс (burst 3) — RR болса 2+1 бө…Далее 12:00
3. Қорытынды және келесі қадамдар • Егер қабылданған процестер мен system/user бөлу дұрыс болса — жоғарғы нәтижелер дәл болады. • Егер сенің нақты бөлінуің басқа (мысалы, P2 мен P3 system, P1 user) немесе әр queue-ға басқа саясат (мысалы system = FCFS, user = RR, quantum=2) қажет болса — маған сол бөліністі немесе квантты айт — мен сол бойынша қайта есептеп, жаңартылған Gantt пен кестені беремін. • Қайта есептеуді репода автоматты түрде шығаруды да қоса тіркей аламын (мысалы, скриптті параметр ретінде қабылдайтын етіп).