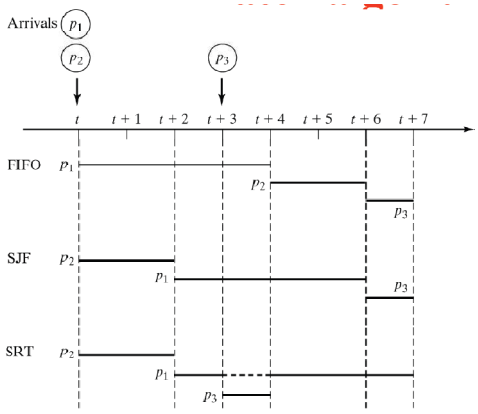
4. Schedulering



OS skal finde ud af, hvornår ready processer skal køre.

Scheduler

1. **Generel struktur**: Find proces med højeste prioritet, find fri cpu, alloker på denne.
2. **Decision mode**: hvornår skal scheduler kaldes?
    a. Preemptive (running processer kan fratages CPUen – scheduleres når en ny proces laves, eks. Vækkes, tidskvantum opbrugt eller når en prio for en ready_a er højere end nuværende) – mere kostbar end nonp…
    b. Nonpreemptive (running processer kan ikke fratages CPUen – scheduleres når proces terminerer, proces blokerer sig selv eller en ny proces finder en ledig CPU) – ikke brugbar i real-time og time-shared.
3. **Prioritetsfunktion** (udregnes ud fra en del parametre; att. service time, real-time in system, total service time, deadline, osv.)
4. **Arbitrationrule:** 2 processer samme prioritet: Random, kronologisk (FIFO).

Scheduleringsalgoritmer

1. **Batchsystemer** (prioriteter flest mulige jobs)
    a. SJF (shortest job first, nonpreemptive)
2. **Interaktivesystemer** (svare requests hurtigt)
    a. RR – round robin eksempel: alle processer har et quantum q som de får lov at køre ud. Når de har brugt deres quantum kommer en anden proces til. Når de alle har brugt deres quantum bliver det reset. (preemptive)

    Problem: ikke prioritering af I/O
3. **Realtidssystemer** (flest mulig jobs bliver ordnet til tiden)
    a. EDF (earliest deadline first, preemptive)

Scheduling algorithm	Decision mode	Priority function	Arbitration rule	
FIFO	Nonpreemptive	r	Random	Batch systemer
SJF	Nonpreemptive	$-t$	Chronological or random	
SRT	Preemptive	$-(t-a)$	Chronological or random	
RR	Preemptive	0	Cyclic	
ML	Preemptive Nonpreemptive	e (same)	Cyclic Chronological	Interaktive systemer
MLF	Preemptive Nonpreemptive	$n - \lfloor lg_2(a/T+1) \rfloor$ (same)	Cyclic Chronological	
RM	Preemptive	$-d$	Chronological or random	Real-tids systemer
EDF	Preemptive	$-(d - r\%d)$	Chronological or random	

a – attained service time
r – real time in system
t – total service time
d – period
e – external priority
n – number of priority levels
T – maximum time at highest level

dOpSys-Linux

1. Vælges efter goodness.
2. Deler op i realtime og regulær prioriteter: realtime har altid højere goodness end regulære.
3. Realtime:
    a. Goodness = Prioritet + 1000. Enten FIFO eller RR.
4. Regulær:
    a. Alle processer får et base quantum når de bliver lavet
    b. Goodness = Base quantum + remaining quantum hvis remaining quantum > 0
    c. Remaining quantum tælles ned når processen får CPU tid.
    d. Når alle processer er løbet tør for remaining quantum, eller er blokkerede starter en ny epoke. Her bliver deres remaining quantum sat til Basequantum + ½ * remaining quantum.
    e. Dermed prioriteteres I/O processer der har været blokkerede da de har et større remaining quantum.
5. Husk at kunne sammenligne med vores implementation!