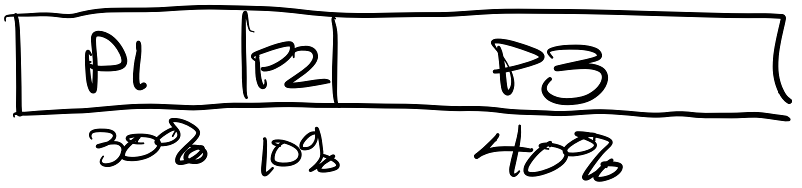
Proportional Sharing of CPUs

Nugget: suppose you want P_1 to run 2x P_2? ie Assign relative priorities that indicate how much each process to run (share of CPU).

Prob of running	
P1	30 %
P2	10 %
P3	40 %



Imagine tickets distributed amongst processes according to desired relative access to CPU.

P_1	30 tix
P_2	10 tix
P_3	40 tix
	100 tix

pick from these 100 tix. The ticket holder gets scheduled next.

Tickets are unitless... just relative values.

P_1 3 tix
P_2 1 tix
P_3 4 tix
10 $\longrightarrow$ pick 1 of 10

P_1 9 tix
P_2 3 tix
P_3 12 tix
24 $\longrightarrow$ pick 1 of 24

All the above work out the same.

How to implement

Any method that enumerates the tickets is a bad idea. Not scalable

Sorting on probability is a good idea BUT also not really that

scalable. More processes means more sorting.
Perhaps a fancy data structure supporting fast insertion & fast finding. Still not good for performance.

Algorithms based on stochastic (random) methods are usually easy to implement. Yay.

They suffer from short term problems but settle to correct performance over time.

Suppose P_n has 1% chance of running. Over any short interval, P_n might not run at all (starve) but over Long periods, P_n will get its "right" share.

Stride Scheduler

a ticket based scheduler w/no randomness, no fancy data structures.

	tix		stride
A	100		100
B	50	$\Rightarrow$	200
C	250		40

$$\text{stride} = \frac{\text{tix}}{\text{bignum}} \qquad \uparrow \ \frac{\text{tix}}{10{,}000}$$

"pass" is an integer. at start of alg, all <u>known</u> processes get pass == $\varnothing$.

1 pick P with lowest PASS.

2 If tie, pick any.

3 At end of quantum, add stride to pass

4 goto 1

Pass A stride 100	Pass B str. 200	Pass C str 40	Run
∅	∅	∅	A
100	∅	∅	B
100	200	∅	C
100	200	40	C
100	200	80	C
100	200	120	A
200	200	120	C
200	200	160	C
200	200	200	etc

Original fix

A	100	2x	ran 2x
B	50	1x	ran 1x
C	250	5x	ran 5x

CFS completely fair scheduler
(Google)

tldr; it aint completely fair.
so they added more.

and more.

and more.

This is typical of Google APIs.
They can take ANYTHING & turn
it into shit.