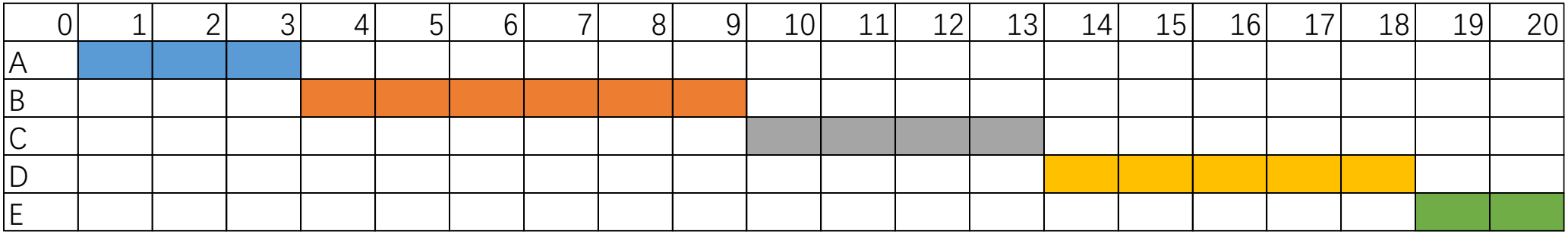
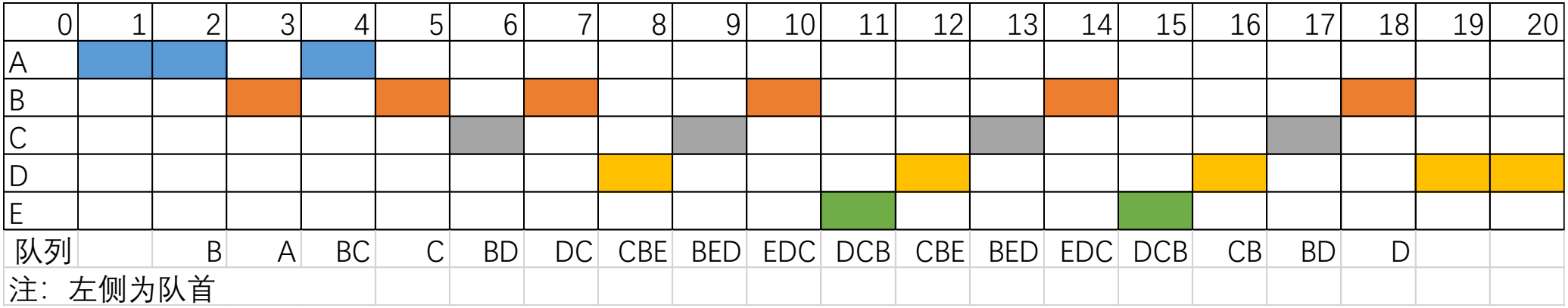
FCFS 先来先服务 非抢占



Round Robin 轮转算法 抢占 (q=1)

注：每过时间片，①先把处于队首的进程放入；②然后，判断被抢占进程是否已经执行完了！若未执行完，将被抢占的进程放入队尾；执行完了就不用放入了！③最后再检查是否有新进程就绪；



队列		B	A	BC	C	BD	DC	CBE	BED	EDC	DCB	CBE	BED	EDC	DCB	CB	BD	D		

注：左侧为队首

Round Robin 轮转算法 抢占 (q=4)

注：每过时间片，①（先上提）先把处于队首的进程放入；②（再回收）然后，判断被抢占进程是否已经执行完了！若未执行完，将被抢占的进程放入队尾；执行完了就不用放入了！③（最后检查新的）最后再检查是否有新进程就绪；

注意队列的顺序，老师画的是队首在右侧，我画的是在左侧

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
队列		B	B	C	C	CD	CD	DBE	DBE	DBE	DBE	BE	BE	BE	BE	ED	ED	D	D	
注：左侧为队首																				

Shortest Process Next(SPN) 最短进程优先 非抢占

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
候选		B-6	B-6	C-4	C-4	C-4	C-4	C-4	C-4	C-4	C-4	D-5	D-5	D-5	D-5					
						D-5	D-5	D-5	D-5	D-5	D-5									
								E-2	E-2											

Shortest Remaining Time(SRT) 最短剩余时间 抢占

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
候选	A剩2	A剩1																		
		B剩6	B剩6	B剩5	B剩5	B剩5	B剩5	B剩5	B剩5	B剩5	B剩4	B剩3	B剩2	B剩1						
				C剩4	C剩3	C剩2	C剩1													
						D剩5	D剩5	D剩5	D剩5	D剩5	D剩5	D剩5	D剩5	D剩5	D剩5	D剩4	D剩3	D剩2	D剩1	
								E剩2	E剩1											

Highest Response Ratio Next(HRRN) 最高响应比优先 非抢占 $R=\frac{w+s}{s}$

注：非抢占，所以，在一个进程执行完/被打断后，才决定下一步要调度谁。【询问，刚一进来的时候，w=0 吗？】

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
R(B)		(0+6)/6	(1+6)/6																	
R(C)				(0+4)/4	(1+4)/4	(2+4)/4	(3+4)/4	(4+4)/4	(5+4)/4											
R(D)						(0+5)/5	(1+5)/5	(2+5)/5	(3+5)/5	(4+5)/5	(5+5)/5	(6+5)/5	(7+5)/5	(8+5)/5	(9+5)/5					
R(E)								(0+2)/2	(1+2)/2	(2+2)/2	(3+2)/2	(4+2)/2	(5+2)/2							

Feedback(q=1) 反馈 抢占

注：每过时间片，①（先上提）先把处于队首的进程放入（对应地这个进程就不在当前队列了！若当前队列还剩，先将其 copy 入当前时间片）；②（再回收）然后，判断被抢占进程是否已经执行完了！若未执行完，将被抢占的进程放入下一优先级 RQ；执行完了就不用放入了！③（最后检查新的）最后再检查是否有新进程就绪；

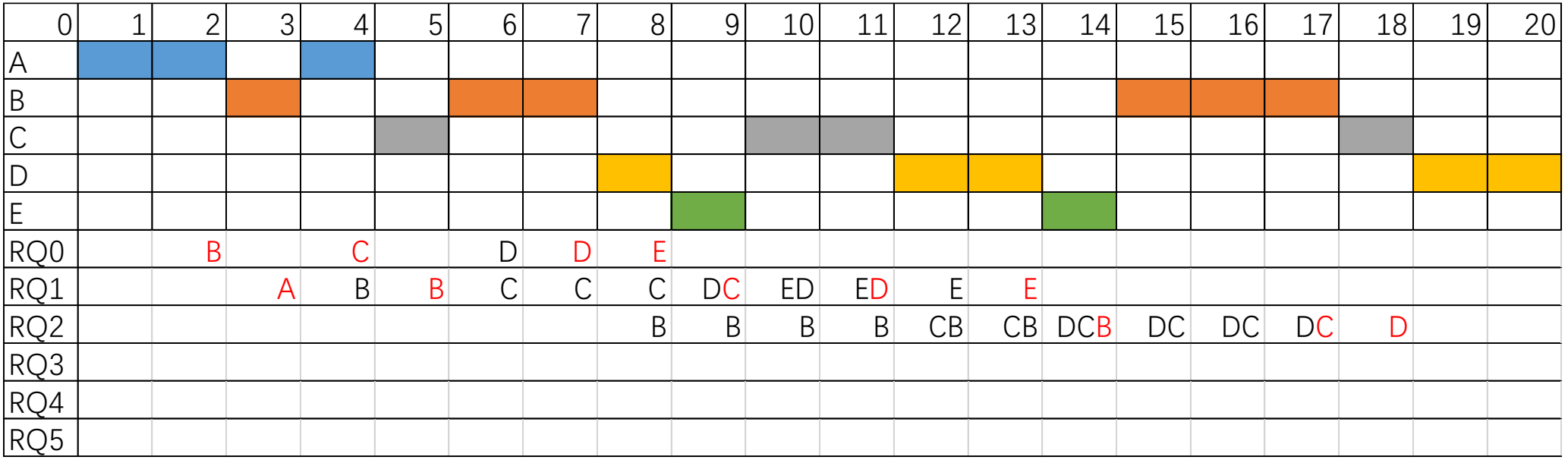
注：当进程从队列中提出来后，就不在队列当中了（从第 2 个时间片到第 3 个时间片，B 从 RQ0 中提走，RQ0 就空了！所有 RQ 中只剩 RQ1 中的 A，所以下一时间片给 A）注意队列的顺序，老师画的是队首在右侧，我画的是在左侧

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
RQ0		B		C		D		E												
RQ1			A	B	B	C	C	D	D	E										
RQ2							B	B	BC	BC	BCD	CD	D							
RQ3													B	BC	CD	D				
RQ4																B	B	D		
RQ5																			B	

Feedback(q=2^i) 反馈 抢占

注：每过时间片，①（先上提）先把处于队首的进程放入，并判断所占时间片大小（毕竟 q=2^i 么）（对应地这个进程就不在当前队列了！若当前队列还剩，先将其 copy 入当前时间片）；②（再回收）然后，判断被抢占进程是否已经执行完了！若未执行完，将被抢占的进程放入下一优先级 RQ；执行完了就不用放入了！③（最后检查新的）最后再检查是否有新进程就绪；

注意队列的顺序，老师画的是队首在右侧，这一把画的和老师的一致



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																					
B																					
C																					
D																					
E																					
RQ0			B		C		D	D	E												
RQ1				A	B	B	C	C	C	DC	ED	ED	E	E							
RQ2									B	B	B	B	CB	CB	DCB	DC	DC	DC	D		
RQ3																					
RQ4																					
RQ5																					

下面来一点 Linux 0.11 中的调度… $counter = \frac{counter}{2} + priority$，基于优先级排队的调度策略