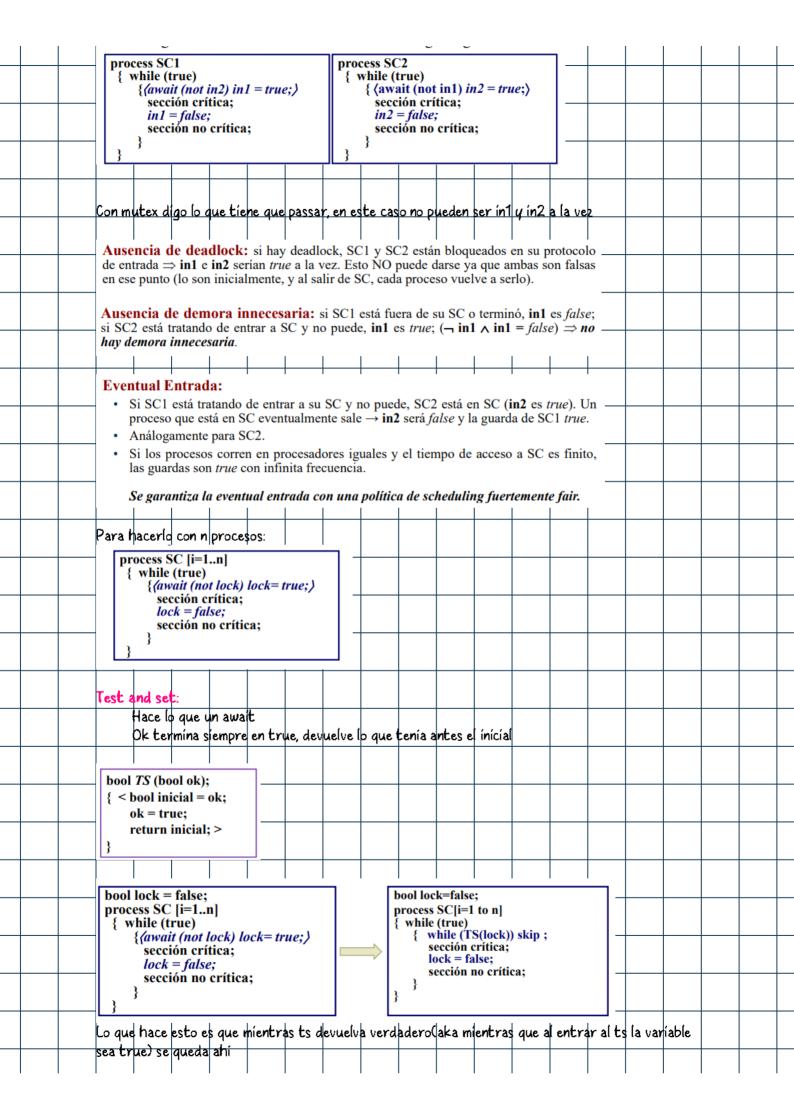
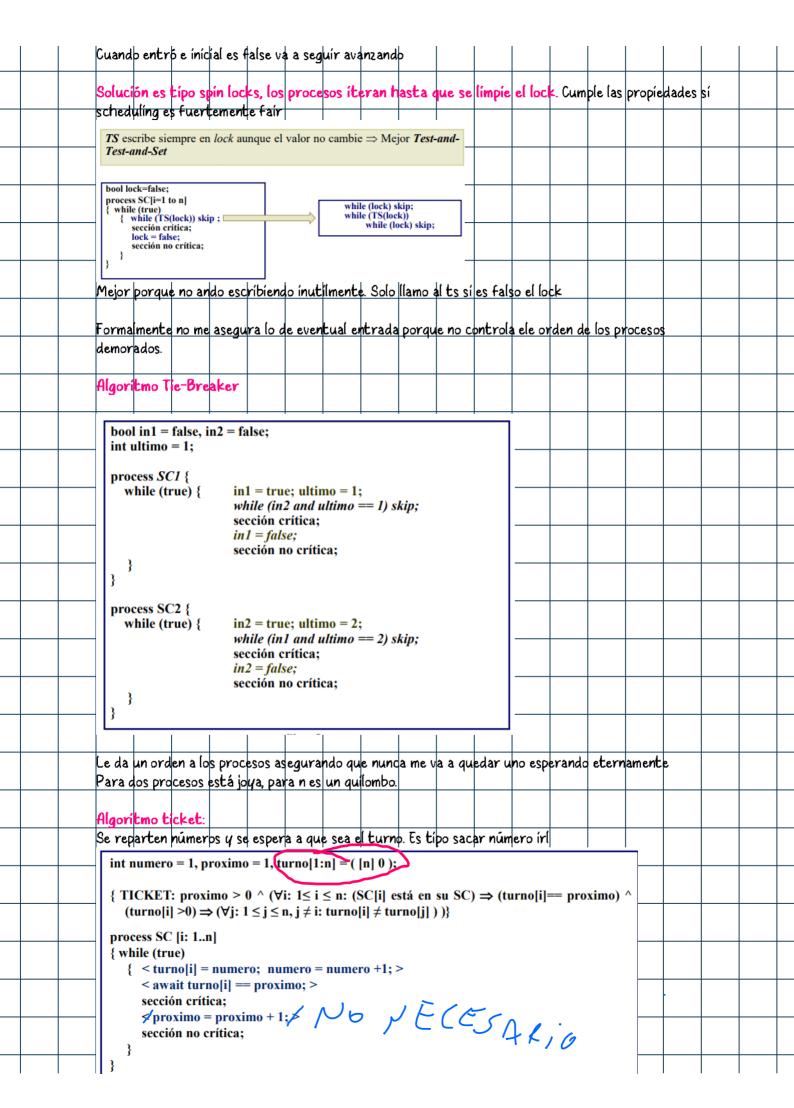
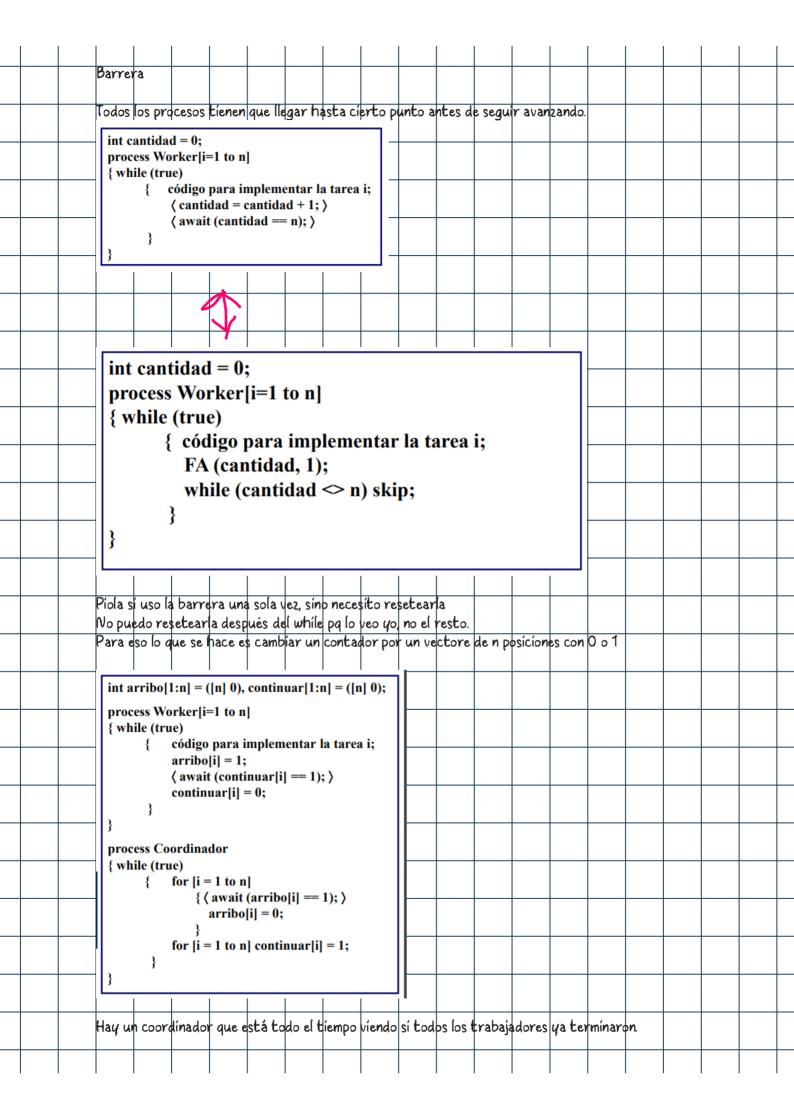
	Clas	e 3																
	jueves	, 29 de	agosto	de 202	4 1	3:11												
		cess S0 while (C[i=1 t	to n]]—										_
	_	{ pr	rotocol			-												
			<i>ección</i> rotoco		-	⇔ ; ⇔												
		_	ección															
	}	,																_
+	Hau a	ue imal	ement	ar alac	aue n	n Derm	ita au	e 2 or	10000	estén	en la o	arte c	ritica	a la ue:	,			
			o hago						, 00,00	0,500.1	J., 1.2 P		10.00	1.3 10.				_
			que de					1.2	L	1.0	4.7							
	1	1	ión mu cía de		1	, ,	'	1		1	1	1	1	ndose.				
	•	Ausen	cía de	demoi	ra inne	cesar	a: sí h	ay un 1	roces	que (quiere	usar s	ección	critica				
			o eject o tengo															_
_			orque ual ent					daba o	odar a	atrar	No se	nuada	nrobas		eta na	a		_
	1	1	a meno	1	_		1			1	1	r	г		1			
	• Cual	guier so	lución al	probler	na de la	SC se pi	uede usa	r para in	nplemen	tar una								
			ca incon															
	• Para		ión atóm ter ; whi															_
			y B cum	71	, ⟨await	(B);⟩ pu	ede impl	lementar	se por n	nedio								
	de =	> while (not B) sk	ip;														
	Es cor		pero no	efícíe	nte po	rque u	n proc	eso es	tá ent	rando	y saliei	ndo de	la sc l	nasta d	que un	•		
	1	1	en el 1	medío a	a míme	sacar	del pr	ocesad	lor lo	que pe	rmite a	ue oti	o proc	eso en	itre y	ejecute	·	
	• Par		ir <i>conte</i>															_
		SCI	Enter ; v	vhile (no	ot B) {S	CExit; I	Delay; S	CEnter	;} S; S	CExit;								
		on x ha icienta	rdwar	e: no p	ermiti	r inter	rupcio	nes										
		'	i=1 to	nl ʃ	l	l	I	l	l		I	l	1 1 ——					_
	-	hile (tr	ue) {				щ		1. 1.									
		seco	habilita ción crí	tica;				protoco										_
	\parallel		ilitar iı ión no			s;	#	protoco	olo de s	alida								
	}																	
	Con u	a máq	uina de	un so	lo prod	esado	r (en n	nultip 1	no and	a)								_
\rightarrow																		 _





```
int numero = 1, proximo = 1, turno[1:n] = ([n] 0):
      { TICKET: proximo > 0 ^ (\forall i: 1 \le i \le n: (SC[i] está en su SC) \Rightarrow (turno[i]== proximo) ^
            (\text{turno}[i] > 0) \Rightarrow (\forall j: 1 \le j \le n, j \ne i: \text{turno}[i] \ne \text{turno}[j]))
      process SC [i: 1..n]
      { while (true)
             { < turno[i] = numero; numero = numero +1; >
                   < await turno[i] == proximo; >
                   <a wait turnoing sección crítica; 
proximo = proximo + 1; 
PO PECESALIG
Podria ser una variable local a cada proceso.
         FA(var,incr): < temp = var; var = var + incr; return(temp) >
        int numero = 1, proximo = 1, turno[1:n] = ([n] \ 0);
        process SC [i: 1..n]
         { while (true)
                 \{ turno[i] = FA (numero, 1); \}
                         while (turno[i] <> proximo) skip;
                         sección crítica:
                         proximo = proximo + 1;
                         sección no crítica;
 fetch and add me permite que cuando alguien más se fija cuanto vale el numero todavia no este
<u>íncrementado pero cuando sale si se incrementa </u>
En el hipotetico caso de que e elecute linfinitamente puede explotar porque nunca las reinicio
Algoritmo bakery
Hace que no haya una varible centralizada con el valor del proximo sino que distribuye el peso entre los
<u>procesos. Cuando en algún momento no se esté ejecutando nada se resetea en O</u>
   int turno[1:n] = ([n] 0);
     {BAKERY: (\forall i: 1 \le i \le n: (SC[i] \text{ está en su } SC) \Rightarrow (turno[i] > 0) \land (\forall j: 1 \le j \le n, j \ne i: j \le n, j \ne n, j 
    turno[j] = 0 \lor turno[i] < turno[j]))
    process SC[i = 1 \text{ to } n]
         while (true)
                            \langle turno[i] = max(turno[1:n] + 1; \rangle
                            for [j = 1 \text{ to n st } j \Leftrightarrow i] \langle \text{await } (\text{turno}[j] == 0 \text{ or } \text{turno}[i] < \text{turno}[j]); \rangle
                            sección crítica
                            turno[i] = 0;
                           sección no crítica
                 }
    }
No puedo hacer lo del mak turno de forma atómical lo que se hace entonces es permitir repetidos
 (sacar los mayor menor al carajo)
Bakerly no usa funciones raras y no me van a explotar las variables. Ineficiente po tiene que recorrer
todo el vector
Me asegura que se cumple eventual entrada con round robin común
```



ı		<u>.</u> 1	1 1	I	1 1	J I	1
	int arribo[1:n] = ([n] 0), continuar[1:n] = ([n] 0);]——					
	process Worker[i=1 to n]						
	{ while (true)						
	{ código para implementar la tarea i; arribo[i] = 1;						
	while (continuar[i] == 0) skip;						
	continuar[i] = 0;						+
	}						
	process Coordinador						
	{ while (true)						
	{ for [i = 1 to n]						
-	{ while (arribo[i] == 0) skip; arribo[i] = 0;	1 MILE	210				-
	}						
	for [i = 1 to n] continuar[i] = 1;						
	}						
		,					
	Un problema es que necesita coordinador. Eso es co	stoso pa ne	cesita un core	solo para él			
	,						
	Tothe and the second						
	Faltan cosítas acá						
	Todo esto es busy waiting.						
	Complejas de implementar						
	Un garrón basicamente						
	Se nedesitan herramientas más potentes> semá	Foros D					
	Se malgasta tiempo de procesadores Mejor estar dormido hasta que me den el procesad	or					
	r lojor oscar sortingo riasga que ine dell'el procesae	T'					
	 						
		. '		. '			•