Simulação manual das soluções do problema de produtores e consumidores

Sistemas Concorrentes - Lista 1 - Exercício 2a

Sistemas Concorrentes - PEL - 2018.2
Anny Caroline Correa Chagas

Enunciado: Simular **manualmente** as várias soluções do problema dos produtores e consumidores com buffer limitado, com semáforos, apresentadas em sala de aula manualmente e através de Redes de Petri. Serão apresentadas as seguintes simulações manuais:

Tabela 1. Solução "AdHoc" - 1 produtor, 1 consumidor e buffer igual a 2

t	Produtor	Consumidor	in	out	Observações
t0			0	0	inicialização
t1	×		0	0	inicia Produtor
t2	×		0	0	`while(1)`
t3	×		0	0	`produz(nextp)`
t4	×		0	0	while falha (`while( (in+1)%N == out)`)
t5	×		0	0	`buffer[in] = nextp`
t6	×		1	0	`in = (in+1)%N`
t7	×		1	0	`while(1)`
t8	×		1	0	`produz(nextp)`
t9	×		1	0	while ok (`while( (in+1)%N == out )`)
t10	×		1	0	`nothing()`
t11	×		1	0	while ok (`while( (in+1)%N == out )`)
t12	×		1	0	`nothing()`
t13		×	1	0	inicia Consumidor
t14		×	1	0	`while(1)`
t15		×	1	0	while falha (`while (in == out)`)
t16		×	1	0	`nextc = buffer[out]`
t17		×	1	1	`out = (out+1)%N`
t18		×	1	1	`while(1)`
t19		×	1	1	while ok (`while(in == out)`)
t20		×	1	1	`nothing()`
t21	×		1	1	`buffer[in] = nextp`
t22	×		0	1	`in = (in+1) % 2`
t23	×		0	1	`while(1)`

Tabela 2. Solução "AdHoc" - 1 produtor, 1 consumidor e buffer igual a 4

t	Produtor	Consumidor	in	out	Obs
t0			0	0	inicialização
t1		×	0	0	inicia Consumidor
t2		×	0	0	while ok (`while (in == out)`)
t3		×	0	0	`nothing()`
t4		×	0	0	while ok (`while (in == out)`)
t5	×		0	0	inicia Produtor
t6	×		0	0	`produz(nextp)`
t7	×		0	0	while falha (`while ((in+1)%N == out)`)
t8		×	0	0	`nothing()`
t9		×	0	0	while ok (`while (in == out)`)
t10	×		0	0	`buffer[in]= nextp`
t11	×		1	0	`in = (in+1)%N`
t12	×		1	0	`while(1)`
t13	×		1	0	`produz(nextp)`
t14	×		1	0	while falhou (`while ((in+1)%N == out)`)
t15	×		1	0	`buffer[in]= nextp`
t16		×	1	0	`nothing()`
t17		×	1	0	while falha (`while (in == out)`)
t18		×	1	0	`nextC = buffer[out]`
t19		×	1	1	`out = (out+1)%N`
t20		×	1	1	`while(1)`
t21		×	1	1	while ok (`while (in==out)`)
t22	×		2	1	`in = (in+1)%N`
t23	×		2	1	`while(1)`
t24		×	2	1	`nothing()`
t25		×	2	1	while falha (`while(in==out)`)
t26		×	2	1	`nextc = buffer[out]`
t27		×	2	2	`out = (out+1)%N`

Tabela 3. Solução com 2 semáforos - 1 produtor, 1 consumidor e TamFila=2

t	Produtor	Consumidor	Item	Vaga	FimFila	IniFila	Obs
t0			0	2			inicialização
t1	×		0	2			inicia Produtor
t2	×		0	2			`FimFila = 1`
t3	×		0	2	1		`while(1)`
t4	×		0	2	1		`produz(V)`
t5	×		0	1	1		`wait(Vaga)`
t6	×		0	1	1		`Fila[FimFila] = V`
t7	×		0	1	2		`FimFila = (FimFila % TamFila)+1`
t8	×		1	1	2		`signal(Item)`
t9	×		1	1	2		`while(1)`
t10	×		1	1	2		`produz(V)`
t11	×		1	0	2		`wait(Vaga)`
t12	×		1	0	2		`Fila[FimFila]=V`
t13	×		1	0	1		`FimFila=(FimFila % TamFila) + 1`
t14	×		2	0	1		`signal(Item)`
t15	×		2	0	1		`while(1)`
t16	×		2	0	1		`produz(V)`
t17	×		2	-1	1		`wait(Vaga)`
t18	Produtor bloqueado para a fila de espera K-Vaga						
t19		×	2	-1	1		inicia Consumidor
t20		×	2	-1	1	1	`IniFila = 1`
t21		×	2	-1	1	1	`while(1)`
t22		×	1	-1	1	1	`wait(item)`
t23		×	1	-1	1	1	`w = Fila[IniFila]`
t24		×	1	-1	1	2	`IniFila = (IniFila % TamFila) + 1`
t25		×	1	0	1	2	`signal(Vaga)`
t26	Produtor desbloqueado						
t27		×	1	0	1	2	`consumir(W)`
t28		×	1	0	1	2	`while(1)`
t29		×	0	0	1	2	`wait(Item)`
t30		×	0	0	1	2	`w = Fila[IniFila]`
t31		×	0	0	1	1	`IniFila = (IniFila % TamFila) + 1`
t32		×	0	0	1	1	`IniFila = (IniFila % TamFila) + 1`
t33	×		0	0	1	1	`Fila[FimFila] = V`
t34	×		1	0	1	1	`signal(Item)`
t35	×		1	0	1	1	`while(1)`
t36		×	1	1	1	1	`signal(Vaga)`
t37		×	1	1	1	1	`consumir(W)`
t38		×	1	1	1	1	`while(1)`
t39		×	0	1	1	1	`wait(Item)`
t40		×	0	1	1	1	`w = Fila[IniFila]`
t41		×	0	1	1	2	`IniFila = (IniFila % TamFila)+1`
t42		×	0	2	1	2	`signal(Vaga)`
t43		×	0	2	1	2	`consumir(W)`
t44		×	0	2	1	2	`while(1)`
t45		×	-1	2	1	2	`wait(Item)`
t46	Consumidor é bloqueado para a fila de espera K-Item						

Tabela 4. Solução com 2 semáforos - 2 produtores, 2 consumidores e TamFila=2 (problemas de concorrência entre os produtores)

t	P1	P2	C1	C2	Item	Vaga	FimFila	IniFila	Obs
t0					0	2			inicialização
t1		×			0	2			inicia P2
t2		×			0	2	1		`FimFila = 1`
t3		×			0	2	1		`while(1)`
t4		×			0	2	1		`produz(V)`
t5		×			0	1	1		`wait(Vaga)`
t6	×				0	2			inicia P2
t7	×				0	2	1		`FimFila = 1`
t8	×				0	2	1		`while(1)`
t9	×				0	2	1		`produz(V)`
t10	×				0	0	1		`wait(Vaga)`
t11	×				0	0	1		`Fila[FimFila] = V`
t12		×			0	0	1		`Fila[FimFila] = V`

A solução que utiliza 2 semáforos apresenta problemas de concorrência. Na Tabela 4, o produtor P2 sobrescreve o valor escrito por P1 na primeira posição do buffer, enquanto que na Tabela 5, os dois consumidores acessam a mesma posição do buffer.

Tabela 5. Solução com 2 semáforos - 2 produtores, 2 consumidores e TamFila=2 (problemas de concorrência entre os consumidores)

t	P1	P2	C1	C2	Item	Vaga	FimFila	IniFila	Obs
t0					0	2			inicialização
t1	×				0	2			inicia P1
t2	×				0	2	1		`FimFila = 1`
t3		×			0	2	1		inicia P2
t4		×			0	2	1		`FimFila = 1`
t5	×				0	2	1		`while(1)`
t6	×				0	2	1		`produz(V)`
t7	×				0	1	1		`wait(Vaga)`
t8	×				0	1	1		`Fila[FimFila] = V`
t9	×				0	1	2		`FimFila = (FimFila%TamFila) + 1`
t10	×				1	1	2		`signal(Item)`
t11		×			1	1	2		`while(1)`
t12		×			1	0	2		`wait(Vaga)`
t13		×			1	0	2		`wait(Vaga)`
t14		×			1	0	2		`produz(V)`
t15		×			1	0	2		`Fila[FimFila ] = V`
t16		×			1	0	1		`FimFila = (FimFila%TamFila) + 1`
t17		×			2	0	1		`signal(Item)`
t18		×			2	0	1		`while(1)`
t19		×			2	0	1		`produz(V)`
t20		×			2	-1	1		`wait(Vaga)`
t21					Produtor 2 é bloqueado para a fila de espera K-Vaga				
t22	×				2	-1	1		`while(1)`
t23	×				2	-2	1		`wait(Vaga)`
t24					Produtor 1 é bloqueado para a fila de espera K-Vaga				
t25				×	2	-2	1		inicia C2
t26				×	2	-2	1	1	`IniFila = 1`
t27				×	2	-2	1	1	`while(1)`
t28				×	1	-2	1	1	`wait(Item)`
t29				×	1	-2	1	1	`w = Fila[IniFila]`
t30				×	1	-2	1	1	`IniFila = (IniFila%TamFila) + 1`
t31			×		1	-2	1	1	inicia C1
t32			×		1	-2	1	1	`IniFila = 1`
t33			×		1	-2	1	1	`while(1)`
t34			×		0	-2	1	1	`wait(Item)`
t35			×		0	-2	1	1	`w = Fila[IniFila]`
t36			×		0	-2	1	2	`IniFila = (IniFila % TamFila) + 1`
t37			×		0	-1	1	2	`signal(Vaga)`
t38					Produtor 1 desbloqueado				
t39			×		0	-1	1	2	`consumir(W)`
t40				×	0	0	1	2	`signal(Vaga)`
t41					Produtor 2 desbloqueado				
t42				×	0	0	1	2	`consumir(W)`

Tabela 6. Solução com 3 semáforos - 1 produtor, 1 consumidor e TamFila=2

t	Produtor	Consumidor	Full	Empty	Mutex	pontP	pontC	Obs
t0			0	2	1	0	0	inicialização
t1		×	0	2	1	0	0	inicia Consumidor
t2		×	0	2	1	0	0	`while(1)`
t3		×	-1	2	1	0	0	`wait(full)`
t4	Consumidor é bloqueado para a fila de espera K-full							
t5	×		-1	2	1	0	0	inicia Produtor
t6	×		-1	2	1	0	0	`while(1)`
t7	×		-1	2	1	0	0	`produz(nextp)`
t8	×		-1	1	1	0	0	`wait(empty)`
t9	×		-1	1	0	0	0	`wait(mutex)`
t10	×		-1	1	0	0	0	`buffer[pontP]=nextp`
t11	×		-1	1	0	1	0	`pontP = (pontP+1)%TamFila`
t12	×		-1	1	1	1	0	`signal(Mutex)`
t13	×		0	1	1	1	0	`signal(Full)`
t14	Consumidor é desbloqueado							
t15	×		0	1	1	1	0	`while(1)`
t16	×		0	1	1	1	0	`produz(nextp)`
t17	×		0	0	1	1	0	`wait(empty)`
t18		×	0	0	0	1	0	`wait(mutex)`
t19	×		0	0	-1	1	0	`wait(mutex)`
t20	Produtor é bloqueado para a fila de espera K-mutex							
t21		×	0	0	-1	1	0	`remove(pontC, nextC)`
t22		×	0	0	-1	1	1	`pontC = (pontC+1)%TamFila`
t23		×	0	0	0	1	1	`signal(mutex)`
t24	Produtor é desbloqueado							
t25	×		0	0	0	1	1	`buffer[pontP]=nextP`
t26	×		0	0	0	0	1	`pontP = (pontP+1)%TamFila`
t27	×		0	0	1	0	1	`signal(mutex)`
t28	×		1	0	1	0	1	`signal(full)`
t29		×	1	1	1	0	1	`signal(empty)`
t30		×	1	1	1	0	1	`consome(nextC)`
t31		×	1	1	1	0	1	`while(1)`
t32		×	0	1	1	0	1	`wait(full)`
t33		×	0	1	0	0	1	`wait(mutex)`
t34		×	0	1	0	0	1	`remove(pontC, nextC)`
t35		×	0	1	0	0	0	`pontC = (pontC+1)%TamFila`
t36		×	0	1	1	0	0	`signal(mutex)`
t37		×	0	2	1	0	0	`signal(empty)`
t38		×	0	2	1	0	0	`consome(nextC)`

Tabela 7. Solução com 3 semáforos - 2 produtores, 2 consumidores e TamFila=2

t	P1	P2	C1	C2	Full	Empty	Mutex	pontP	pontC	Obs
t0					0	2	1	0	0	inicialização
t1		×			0	2	1	0	0	inicia P2
t2		×			0	2	1	0	0	`produz(nextP)`
t3		×			0	1	1	0	0	`wait(empty)`
t4	×				0	1	1	0	0	inicia P1
t5	×				0	1	1	0	0	`produz(nextP)`
t6	×				0	0	1	0	0	`wait(empty)`
t7	×				0	0	0	0	0	`wait(mutex)`
t8	×				0	0	0	0	0	`buffer[pontP]=nextP`
t9		×			0	0	-1	0	0	`wait(mutex)`
t10	P2 é bloqueado para a fila de espera K-mutex									
t11			×		0	0	-1	0	0	inicia C1
t12			×		0	0	-1	0	0	`while(1)`
t13			×		-1	0	-1	0	0	`wait(full)`
t14	C1 é bloqueado para a fila de espera K-full									
t15				×	-1	0	-1	0	0	inicia C2
t16	×				-1	0	-1	1	0	`pontP = (pontP+1)%TamFila`
t17	×				-1	0	0	1	0	`signal(mutex)`
t18	P2 é desbloqueado									
t19	×				0	0	0	1	0	`signal(full)`
t20	C1 é desbloqueado									
t21				×	0	0	0	1	0	`while(1)`
t22				×	-1	0	0	1	0	`wait(full)`
t23	C2 é bloqueado para a fila de espera K-full									
t24			×		-1	0	-1	1	0	`wait(mutex)`
t25	C1 é bloqueado para a fila de espera K-mutex									
t26			×		-1	0	-1	1	0	`buffer[pontP] = nextP`
t27			×		-1	0	-1	0	0	`pontP = (pontP+1)%TamFila`
t28			×		-1	0	0	0	0	`signal(mutex)`
t29	C1 é desbloqueado									
t30			×		0	0	0	0	0	`signal(full)`
t31	C2 é desbloqueado									
t32			×		0	0	0	0	0	`remove(pontC, nextC)`
t33			×		0	0	0	1	0	`pontC = (pontC+1)%TamFila`
t34			×		0	0	1	1	0	`signal(mutex)`
t35			×		0	0	0	1	0	`wait(mutex)`
t36			×		0	0	0	1	0	`remove(pontC, nextC)`
t37			×		0	0	0	0	0	`pontC = (pontC+1)%TamFila`
t38			×		0	0	1	0	0	`signal(mutex)`
t39			×		0	1	1	0	0	`signal(empty)`
t40			×		0	1	1	0	0	`consome(nextC)`
t41			×		0	2	1	0	0	`signal(empty)`
t42			×		0	2	1	0	0	`consome(nextC)`