handout v 1	V06b Cheng Tan, CS3650 2/14/24. 11:12AN Week 6.b	I handout w	06b Cheng Tan, CS3650	2/14/24. 11:12 AM
2	1. Producer/consumer example:	58 59	2. Producer/consumer revisited [also known as bounded buffer]	
3	1. Producer/consumer example:	60	2a. Producer/consumer [bounded buffer] with mutexes	
5	/*	61	Za. Froducer/consumer [bounded burrer] with matexes	
6	"buffer" stores BUFFER SIZE items	62	Mutex mutex;	
7	"count" is number of used slots. a variable that lives in memory	63		
8	"in" is next empty buffer slot to fill (if any)	64	void producer (void *ignored) {	
9	"out" is oldest filled slot to consume (if any)	65	for (;;) {	
10	*/	66	<pre>/* next line produces an item and puts it in nextProduced */</pre>	
11		67	<pre>nextProduced = means_of_production();</pre>	
12	void producer (void *ignored) {	68		
13		69	acquire(&mutex);	
14 15	for (;;) {	70	<pre>while (count == BUFFER_SIZE) {</pre>	
15	<pre>/* next line produces an item and puts it in nextProduced */ nextProduced = means_of_production();</pre>	71	release(&mutex); yield(); /* or schedule() */	
10	while (count == BUFFER_SIZE)	72	acquire(&mutex);	
18	; // do nothing	74	}	
10	buffer [in] = nextProduced;	75	,	
20	$in = (in + 1) \otimes BUFFER SIZE;$	76	<pre>buffer [in] = nextProduced;</pre>	
21	count++;	1 77	$in = (in + 1) $ BUFFER_SIZE;	
22	}	78	count++;	
23	}	79	release(&mutex);	
24		80	}	
25	void consumer (void *ignored) {	81	}	
26	for (;;) {	82		
27	while (count $== 0$ )	83	<pre>void consumer (void *ignored) {</pre>	
28	; // do nothing	84	for (;;) {	
29	<pre>nextConsumed = buffer[out]; nut = (sut = 1) 0 PUFFFP STAFE</pre>	85		
30 31	<pre>out = (out + 1) % BUFFER_SIZE; count +</pre>	86	acquire(&mutex);	
32	count; /* next line abstractly consumes the item */	87	<pre>while (count == 0) {     release(&amp;mutex);</pre>	
33	consume_item(nextConsumed);	89	yield(); /* or schedule() */	
34	}	90	acquire(&mutex);	
35	}	91	}	
36	-	92		
37	/*	93	<pre>nextConsumed = buffer[out];</pre>	
38	what count++ probably compiles to:	94	out = (out + 1) % BUFFER_SIZE;	
39	reg1 < count # load	95	count;	
40	reg1 < reg1 + 1 # increment register	96	release(&mutex);	
41	count < reg1 # store	97		
42 43	what sound compile to:	98	<pre>/* next line abstractly consumes the item */ consume item(nextConsumed);</pre>	
43 44	<pre>what count could compile to:     req2 &lt; count # load</pre>	100	<pre>consume_item(nextConsumed); }</pre>	
44 45	reg2 < count # load reg2 < reg2 - 1 # decrement register	100	}	
45	count < reg2 = 1 # decrement register	101	L	
47	*/			
48				
49 50	What happens if we get the following interleaving?			
51	reg1 < count			
52	reg1 < reg1 + 1			
53	reg2 < count			
54	reg2 < reg2 - 1			
55	count < regl			
56	count < reg2			
57				
	Page 1 of S	·		Page 2 of 3

handout w0	Obb         Cheng Tan, CS3650         2/14/24. 11:12 AM			
103	2b. Producer/consumer [bounded buffer] with mutexes and condition variables			
104				
105	Mutex mutex;			
106	Cond nonempty;			
107	Cond nonfull;			
108				
109	<pre>void producer (void *ignored) {</pre>			
110	<pre>for (;;) {     /* next line produces an item and puts it in nextProduced */</pre>			
111				
112	<pre>nextProduced = means_of_production();</pre>			
113				
114	<pre>acquire(&amp;mutex);</pre>			
115	while (count == BUFFER_SIZE)			
116	cond_wait(&nonfull, &mutex);			
117				
118	<pre>buffer [in] = nextProduced;</pre>			
119	<pre>in = (in + 1) % BUFFER_SIZE;</pre>			
120	count++;			
121	cond_signal(&nonempty, &mutex);			
122	release(&mutex);			
123	}			
124	}			
125				
126	void consumer (void *ignored) {			
127	for (;;) {			
128				
129	acquire(&mutex);			
130	while (count == 0)			
131	<pre>cond_wait(&amp;nonempty, &amp;mutex);</pre>			
132				
133	<pre>nextConsumed = buffer[out];</pre>			
134	out = (out + 1) % BUFFER_SIZE;			
135	count;			
136	cond_signal(&nonfull, &mutex);			
137	release(&mutex);			
138				
139	<pre>/* next line abstractly consumes the item */</pre>			
140	<pre>consume_item(nextConsumed);</pre>			
141	}			
142	}			
143				
144				
145	Question: why does cond_wait need to both release the mutex and			
146	sleep? Why not:			
147				
148	<pre>while (count == BUFFER_SIZE) {</pre>			
149	release(&mutex);			
150	cond_wait(&nonfull);			
151	<pre>acquire(&amp;mutex);</pre>			
152	}			
153				
154				
	Page 3 of 3			
	rage s of s			