



Class Exercise

When can/do we switch processes (or threads)?

Process Switching

We could switch processes any time the OS has control, i.e.,

- Interrupt occurs
 - Clock
 - I/O interrupt
 - Page fault
- Trap occurs
 - Trace
 - Protection fault
- System call
 - I/O request
 - Wait for child
 - ► etc.



Process Switch Overheads



Process Switch Overheads

To switch processes, system must Save the state of the old process Load the saved state for the new process

To switch processes, system must

- Save the state of the old process
- Load the saved state for the new process

The Essence of Scheduling



Scheduler manages some of these state transitions:



Scheduling Goals



Scheduling Goals

What are some possible goals for a scheduler? What could we try to optimize?

Many different scheduling algorithms

- Tradeoffs
- Different goals \Rightarrow Different choices

What are some possible goals for a scheduler?

What could we try to optimize?

Scheduling Exercise



Consider t	ne tollow	ing set c	of running process
	Arrival	Burst	Priority
Process	Time	Time	(if applicable)
Α	0	10	3
В	0	1	1
С	0	2	3
D	0	1	4
E	0	5	2

Consider the following set of running processes

Example Answer

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19



Example Answer

	A			A		A		A		A									
																		E	3
		C																	
D																			
												E			E		Е		
0 1	2	3 4	5	6	7	8	9	10	11	12	1	3 1	41	5	16	1	71	8	19
D	А	C		А		А		А		А		E			Е		Е	E	3



	A		Α	A	A	Å			
		Ċ							
P									
	-	-	-			-	6	£	E

This is a priority-based round-robin scheduler with a 2-second (well, 2-unit) time slice, where higher numbers are higher priorities. All processes arrive at time 0, so D runs first (priority 4). Then A and C alternate; C quickly finishes so A hogs the CPU until it's done. Then E runs exclusively, followed by B.

First Come, First Served





A		
	B	
	C	
		D

We assume that although the processes all arrive at time 0, they arrive in alphabetical order. Simple.

Shortest Burst First

		A
B C		
	E	
0 1 2 3 4 B D C	F 5 6 7 8 9	A



Within each burst leave, it's FCFS.

Shortest Burst First

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

B C

Nonpreemptive Priority



Class Question

Should low-priority jobs starve?



									A						
8															
															c
					-	-	-	-	-	-	-	-	-	-	-
	-		-	4											
0 1	2	14	\$	4	ż	4	2	10	ņ	12	13	14	15	16	12
		5		Т					A					Т	c

Note that here, **low** numbers mean high priority. Urgh! So we run B, then E, then A, C, D in that order. It's FCFS sorted by priority.

Round Robin

																			4
A					A			A		A		A		А	A	A	A	А	
	В																		
		С				C]												
			D																
				Е			E		E		Ε		E						
0 1		2 3	3 4	4 5	5 6	5 7	7 8	3 9	91	0 1	1 1	2 1	3 1	41	51	61	71	81	9
Α	В	С	D	Е	Α	C	E	Α	E	Α	Е	Α	E	А	А	А	Α	Α]

heduling



OS/161 MLF Scheduler

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25



MLF == Multi-Level Feedback. Three parameters are summed: base priority (niceness, -256 to 256); dynamic priority (decremented by delta of 16 when stopped by a clock interrupt, incremented by 16 when blocks or yieds); and compensation priority (set to 0 when scheduled, incremented whenever passed over).



Time = 0

 Octol MLF Scheduler

 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1

OS/161 MLF Scheduler

0	1	2	3	4	5	6	7	8	9	10) 1	1 1	2	13	14	15	16	17	18	19	20	21	22	23	24	25



Initially A is the only process, so it is chosen to run (boldface).

Priority	Α	В	С	D	E
Base	32				
Dynamic	0				
Compensation	0				
Total	32				

Time = 0

15/52

OS/161 MLF Scheduler

	A																								
°	1 A	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25





A begins running. It has a 2-unit time slice.

OS/161 MLF Scheduler

þ	A																								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Γ	A																								

Priority	Α	В	С	D	Е	
Base	32	2		0		
Dynamic	0	0		0		Time = 1
Compensation	0	0		0		
Total	32	2		0		



B and D arrive, with different base priorities.

OS/161 MLF Scheduler

Ľ	A																								
			2		į	2	,			10		12	12	14	15	16	17	10	10	20	21	22	22	24	25
ľ	A	Ť	ì	1	ì	Î	ĺ.	ì	1	ľ	ï	12	-ip	14	i j	10	ľ	10	19	20	21	22		24	1

CS34	OS/161 MLF Scheduler
Scheduling	Image: Second control of the second control

A's first timie slice expires, so its dynamic priority is reduced. But it's still highest.

Priority	Α	В	С	D	Е
Base	32	2	2	0	
Dynamic	-16	0	0	0	
Compensation	0	0	0	0	
Total	16	2	2	0	

Time
$$= 2$$

OS/161 MLF Scheduler

Ľ	A																								
			2		į	2	,			10		12	12	14	15	16	17	10	10	20	21	22	22	24	25
ľ	A	Ť	ì	1	ì	Î	ĺ.	ì	1	ľ	ï	12	-ip	14	i j	10	ľ	10	19	20	21	22		24	1

Priority	Α	В	С	D	Е	
Base	32	2	2	0		
Dynamic	-16	0	0	0		Time = 2
Compensation	0	1	1	1		-
Total	16	3	3	1		



A continues to run. Everybody else gets compensation.

OS/161 MLF Scheduler

È	A		A																						
Ŷ.	1	2	3	4	5	6	Ż	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Γ	А	Т	А																						



	schee	luler				
		* 11 1	2 12 14 1	097		0.11 II II II II
Priority Base	A 32	8	C 2	0	E	
Dynamic Compensation	-16	0	0	0	0	Tim
Total	16					

E arrives at time 3. A still has the highest priority. We don't compensate because a time slice didn't end, so we didn't reschedule.

Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-16	0	0	0	0
Compensation	0	1	1	1	0
Total	16	3	3	1	8

OS/161 MLF Scheduler

h	A	Ť	A	1																					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	À		A																						

5-17	;34
⊂	– Scheduling
2013-05	OS/161 MLF Scheduler

A's time slice ends, so we reduce its dynamic priority. Time to reschedule!

S/161 MLF Schedule

ABCDE

Priority	Α	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	0	0	0	0	
Compensation	0	1	1	1	0	
Total	0	3	3	1	8	

Time
$$= 4$$

OS/161 MLF Scheduler

h	A	Ť	A	1																					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	À		A																						

	ÇS34
5-17	Scheduling
13-0	OS/161 MLF Scheduler
20	

A A							
1000							
- Contraction of the local division of the l							
111111	111	* 11	9.0.04.1	5 16 17	99	30 31	22 21 24 2
Priority	A	В	С	D	Ε		
Base	32	2	2	0	8		
Dynamic	-32	0	°	0	0		Time
Completellion		× .	~	× .	0		

E now has the highest priority. Everybody else (including A) gets compensation.

Priority	Α	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	0	0	0	0	
Compensation	1	2	2	2	0	
Total	1	4	4	2	8	

Time
$$= 4$$

OS/161 MLF Scheduler

t	A		A																					
			_																					
		_		_																				
		T	Г	E																				
o	1	2	3	4 :	56	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
t	A		A	E	<u> </u>												_							

CS34 └──Scheduling ºo ºo ºo OS/161 MLF Scheduler

the second se					
	 * 11 12	9.91	n 19 12	 10 JI JI	

E runs for one time unit and then blocks for I/O. It gets a 16-point dynamic priority boost for doing I/O, but isn't eligible for scheduling because it's blocked.

Priority	Α	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	0	0	0	16	
Compensation	1	2	2	2	0	
Total	1	4	4	2	24	

Time
$$= 5$$

OS/161 MLF Scheduler

Ľ	A	Ť	A																									
					E	 				 							_											
0 L	1	2	3	4	5	5	ł	8	9	10	1	1 1	2	13	1	4	15	16	1	71	8	19	20	21	22	23	24	25
	А		А		Е																							



0	8/16	1 N	/LF	S	he	dule					-	
			1			10 11	9.9	9 Q	14.5		30 JI -	12 JU 24 2
	Prior	ty	_	_	A 32	8		2	0	E		
	Dyna Com	pens	ation		-32	0		3	03	16 0		Time
	Treat				2	2		ε		24		

B and C are now tied for the highest total, so we arbitrarily choose B. B's compensation gets set to 0, and everybody else who is passed over has their compensation incremented. Note that E doesn't get compensation because it is blocked for I/O, so it wasn't passed over.

Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-32	0	0	0	16
Compensation	2	0	3	3	0
Total	2	2	5	3	24

OS/161 MLF Scheduler

t	A		A																								
						В	[
		_	_	_																							
					E																						
ò	1	2	3	4	5	5 6	5 3	7 8	8	9	10	1	1 1	2	13	14	15	16	17	18	19	20	21	22	23	24	25
	Α		Α		Е	В																					

S/161 MLF S	Sched	uler				
* * *						
1						
1						
11111	111	* 1 9	9.94	5 16 17	18 19 20	11 12 13 24 25
A A I						
Pricety	A	в	C	D	E	
	32	2	2	0	8	
Ease			_	0	16	Time =
Dynamic	-32	16	0			
Base Dynamic Compensation	-32 2	16	3	3	0	

B runs for one unit and blocks for I/O. It gets a dynamic boost of 16. C now has the highest priority.

Priority	А	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	16	0	0	16	
Compensation	2	0	3	3	0	
Total	2	18	5	3	24	

Time
$$= 6$$

OS/161 MLF Scheduler



2013-05-17	CS34 Scheduling OS/161 MLF Scheduler

	Image: Control of the state of the								
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1								
	Image: Control of the state of the	and the second s							
	I I	and the second second							
Protey A S C D E	Image: 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	1							
	Priority A B C D E Base 32 2 0 5			10 11 12	10.14.1	1 14 15		20 21 2	
Priority A B C D E	Priority A B C D E Base 32 2 2 0 5	A A E E							
Priority A B C D E	Base 32 2 2 0 5								
	Base 32 2 2 0 8	Priority	A	в	c	D	E		
Dynamic -32 16 0 0 16 Time		Dynamic	-32	16	0	0	16		Time
Dynamic -32 16 0 0 16 Time Compensation 3 0 0 4 0	Compensation 3 0 0 4 0	Dynamic Compensation	-32 3	16	0	0 4	31 0		Time

C is chosen to run (boldface) and has its compensation set to 0; everybody non-blocked gets a compensation bump.

Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-32	16	0	0	16	
Compensation	3	0	0	4	0	
Total	3	18	2	4	24	

Time
$$= 6$$

OS/161 MLF Scheduler



1 1	-	C					
C 1 1 4 4 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		C					
1 1							
Priority A B C D E Base 32 2 2 0 5	1						
A A B C D E Base 32 2 2 0 5	111111	1111	19.11.1	1999	5 16 10	***	0 31 32 28 24 2
Priority A B C D E Base 32 2 2 0 8		c					
Base 32 2 2 0 8				0	0	5	
	Priority	A	D	-			

B's I/O is finished, and C blocks for I/O. C gets a dynamic boost of 16. B's numbers don't change; they were handled when it blocked.

Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-32	16	16	0	16	
Compensation	3	0	0	4	0	
Total	3	18	18	4	24	

OS/161 MLF Scheduler



	ÇS34
-17	Scheduling
2013-05	OS/161 MLF Scheduler

B runs again (as a reward for having done I/O). A and C get compensation.

S/161 MLF Schedule

ABCDE

Priority	А	В	С	D	E
Base	32	2	2	0	8
Dynamic	-32	16	16	0	16
Compensation	4	0	0	5	0
Total	4	18	18	5	24

Time
$$= 7$$

OS/161 MLF Scheduler



Priority	А	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	32	16	0	16	
Compensation	4	0	0	5	0	
Total	4	34	18	5	24	



B again blocks, getting another dynamic boost. D will run next.

OS/161 MLF Scheduler



Priority	А	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	32	16	0	16	
Compensation	5	0	0	0	0	
Total	5	34	18	0	24	



D runs, so its compensation gest set to 0. A gets more compensation.

OS/161 MLF Scheduler



113-05-17	CS34 - Scheduling - OS/161 MLF Schedule
2013	

We skip to time 10, when D's time slice runs out. It gets a dynamic penalty. A will run next.

DS/161 MLF Schedule

ABCDE

Time = 10

Priority	А	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	32	16	-16	16	
Compensation	5	0	0	0	0	
Total	5	34	18	-16	24	

OS/161 MLF Scheduler



Time = 10

32/52

Priority	Α	В	С	D	Е	
Base	32	2	2	0	8	
Dynamic	-32	32	16	-16	16	
Compensation	0	0	0	1	0	
Total	0	34	18	-15	24	



A is chosen to run; its compensation is reset. D gets compensation.

OS/161 MLF Scheduler



Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-48	32	16	-16	16	
Compensation	0	0	0	1	0	
Total	-16	34	18	-15	24	



A expires its time slice. It becomes (barely) lower priority than D.

ABCDE

Time = 12

OS/161 MLF Scheduler



Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-48	32	16	-16	16	
Compensation	1	0	0	0	0	
Total	-15	34	18	-16	24	



D is chosen to run; A gets compensation.

0S/161 MLF Scheduler

ABCDE

Time = 12

OS/161 MLF Scheduler



Priority	А	В	С	D	E
Base	32	2	2	0	8
Dynamic	-48	32	16	-32	16
Compensation	1	0	0	0	0
Total	-15	34	18	-32	24



D uses up its slice. A and D are now round-robining.

OS/161 MLF Scheduler



Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-48	32	16	-32	16
Compensation	0	0	0	1	0
Total	-15	34	18	-31	24





A runs.

OS/161 MLF Scheduler



Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-64	32	16	-32	16	
Compensation	0	0	0	1	0	
Total	-32	34	18	-31	24	



and the second sec						
A A		· ·		<u> </u>		
	1					
			0			
		_				
1						
111111	1.1.1	10.11.5	10.14	10.10.10	10.19.20	21 22 21 24 25
	C 8 0	6	b	A		
Pricety	A	в	C	D	E	
Base	32	2	2	0	8	
Dynamic	-64	32	15	-32	16	Time -
				1	0	

A expires.

OS/161 MLF Scheduler



Priority	А	В	С	D	E	
Base	32	2	2	0	8	
Dynamic	-64	32	16	-32	16	
Compensation	1	0	0	0	0	
Total	-31	34	18	-32	24	





D runs.

OS/161 MLF Scheduler

Compensation

Total



0

18

0

-48

0

24

0

34

-31



	٠	Т	٨							À	Т		r	A	Т		t.					
	1															1						
	p							0			II.	ò	Ï		ľ	0	5					
			8	ľ																		
t	1	i	1		ŀ	c			1	A 10	Ť	11	Ť	Å	ï	10	ĩ	7	20	1	-12	-
1	Pr	ior	ŧγ	_	_	_	_	A	_	в	_	-		_	D	_	_	E				
ľ	B	134	1				-	2		2			2		0			8				_
	C	2	Dill Dill	C 18.8	601		-6	7		32		5	5		-0							10
	2														Ľ,							

B and C finish I/O at time 17. Properly speaking, they should interrupt D at this point. But either the diagram is wrong, or the MLF scheduler refuses to preempt a running CPU-bound task. (If so, it's not doing well; this would be a good time to discuss deceptive idleness and anticipatory scheduling.)

OS/161 MLF Scheduler

-30

34

Total



19

-47

24



Prio	nty it	A	8	c		E	
114	1 1 1		* 11 1	2 13 14	1 14 17	1 19 20	
	interior de la companya de la			0	•		
		E .					
	A		A		A.		

B is now the highest priority. Compensations are adjusted, and it runs.

OS/161 MLF Scheduler

Compensation

Total

2

50

19

-30



2013-05-17	CS34 CScheduling CS/161 MLF Scheduler

B is heavily I/O-bound, so it blocks yet again, getting another dynamic boost.

S/161 MLF Scheduler

C 0 0 ABCDE

16 -48

-30 50 19 -47 24

Time = 19

2	2	0	8	
48	16	-48	16	
0	1	1	0	

-47

24

OS/161 MLF Scheduler



i noncy		-	•	-	_
Base	32	2	2	0	8
Dynamic	-64	48	16	-48	16
Compensation	3	0	2	2	0
Total	-29	50	20	-46	24



Time = 19

42/52

A.	A I			-	T A	1	À	1	-		
			1								
E			N	1		0		•	1		
		1									
H	A 1	i.	c s	H	1	0	A	0	1	30 11	22 29 2
Pr	orty			A	В	- (-	D	E		
Ba	50			32	2	-		0	8		
Ce	mper	: Isation	, '	3	40	3		2	0		Line

E has a higher base priority than C, so it runs. (Eventually, C's compensation would help it out.)

OS/161 MLF Scheduler



Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-64	48	16	-48	32
Compensation	3	0	2	2	0
Total	-29	50	20	-46	24



E blocks for I/O. C will run now.



OS/161 MLF Scheduler



Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-64	48	16	-48	32
Compensation	4	0	0	3	0
Total	-28	50	18	-45	24



		A		A		
1						
C						
	0			0		
1					1	
	11	* 11 5		0.0.2	1.11.2	0 21 22 20
A A I I C	8 0	A	D	A 0	11	
Printly		8	0	0	-	
Base	32	2	2	0	8	
Dynamic	-64	48	15	-48	32	Tie
Compensation	- 4	0	0	3	0	



Time = 20

44/52

OS/161 MLF Scheduler



Priority	А	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-64	48	32	-48	32
Compensation	4	0	0	3	0
Total	-28	50	34	-45	24



- A						T	À.			À	Т						
												1	P				
												.1					
					•				0			•					
	1	1											4	P	4		
H	11	11	+ c	1	-	Ť	11	1		A	14	17	Ť,	÷	20 :	11	2.2
10	oth		_	_	A .	_	8	_	c	_	0	_	_	E			
Ba	80			3	2		2		2		0			5			
		ic i		-6	4						-48						ιæ

C blocks, getting another 16 dynamic. Now we only have CPU-bound processes left.

OS/161 MLF Scheduler



Priority	Α	В	С	D	E
Base	32	2	2	0	8
Dynamic	-64	48	32	-48	32
Compensation	0	0	0	4	0
Total	-32	50	34	-44	24



1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1			A I		A.		
C C C 1 3 4 7 10 10 10 1 3 4 7 10					1		
0 0 0 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1						4	
$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$				0	0		
I J	1					1	
A A I			* 11 5		0.0.7	9 19 20	
Priority A B C D E Base 32 2 2 0 6 Dynamic -64 46 32 -46 32 T	1 1 1		4	D	A 0		
Base 32 2 2 0 5 Dynamic -64 48 32 -48 32 T	Priority	A	в	c	D	E	
Dynamic -64 48 32 -48 32 T	Base	32	2	2	0	8	
	Dynamic	-64	48	32	-48	32	Tin
	Compensation	•					



Time = 21

46/52

OS/161 MLF Scheduler





A A		À.		A.		A
1	1				1	
	0		0	0		
1					1	
	7 8 8 C 8 0	10 11 1 A	0		3 9 8	21 22 21 C A
Priority	A	в	C	D	E	
Base	32			0	8	
Dynamic	-00	40	32	-40	32	10
Companyation						

A is preempted.

Priority	Α	В	С	D	E
Base	32	2	2	0	8
Dynamic	-80	48	32	-48	32
Compensation	0	0	0	4	0
Total	-48	50	34	-44	24

OS/161 MLF Scheduler



Priority	Α	В	С	D	Е
Base	32	2	2	0	8
Dynamic	-80	48	32	-48	32
Compensation	1	0	0	0	0
Total	-47	50	34	-48	24



		A I		A THE		A
1						
					6	
			0	0		
1					1	
	1 1 1	* 11 1	D 14	1 14 12 A 2	10 10 20 21	12 12 2
Ports						
Base	32	2	2	0	8	
	-80	48	32	-48	32	Tim
Dynamic						
Dynamic Compensation	1	0	0	0	0	



OS/161 MLF Scheduler



i nonty	<i>/</i> \		0			
Base	32	2	2	0	8	
Dynamic	-80	48	32	-64	32	
Compensation	1	0	0	0	0	
Total	-47	50	34	-64	24	



D is preempted. A will run again here.

A A			À.	1	À Ì		1 A	100
		0		0		2		0
1						1		
111	111	1.1.1	0 11 X	2 13 14	0.01	2 10 10 2	0 31 33	2.2.2
1 1 1				D	A			0
ricety		A	в	c	D	E		
3880		32	2	2	0	8		
								There is



What happens when a low-priority thread holds a lock that a high priority thread wants?

Real-Time Scheduling

Consider two applications

- Video playback
- Controlling cancer treatment X-ray

How can we deal with their needs?

CS34 Scheduling

Consider two applications • Video playback • Controlling cancer treatment X-ray How can we deal with their needs?

Two approaches: Hard Real-Time

You do some form of "admission control":

- A process will say "this is what I need in the future"
- Depending on what it's already committed to, the scheduler will say yes or no to that process

Soft Real-Time

- You usually give a deadline ("I want this done by this time")
- If it doesn't get done, it doesn't get done (dropping frames during video playback)
 - You might not notice if only a few operations don't make it

Class Exercise

Should we preempt kernel code, or wait until we hit user code?



In hard real-time, definitely yes.

Traditionally, the kernel was never preemptible. A system call went until it decided to yield.

Should we preempt kernel code, or wait until we hit user co

Advantages to non-preemptible kernel: easier to code; easier to make thread-safe; avoids many race conditions and bugs. Advantages to preemptible kernel: you're worrying about mutithreaded cores anyway; can improve average latency; time spent in kernel could lead to scheduling unfairness; even with fair scheduler that tracks in-kernel time, you could get bad latency; it's possible to have kernel ignore timer interrupts (prevent preemption), while still having other interrupts (like disk access).

But... there are other solutions to latency. Instead of preemption, kernel could explicitly yield during hard/slow operations (original Unix kernel did that).

OS 161 preempts the kernel.

Once you have a multiprocessor, you already need locks and stuff in your kernel, so making it preemptible is not as big a deal.