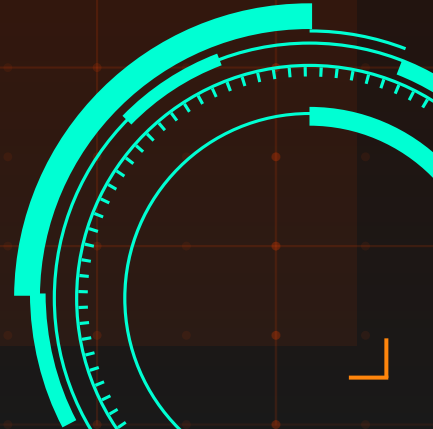
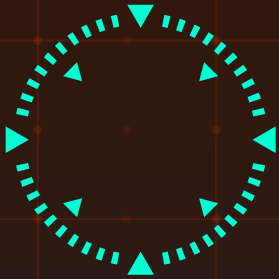


# Task 6: Adding New Schedulers to XV6

---

Jana Saleh 900204192  
Mariam Dahab 900192441  
Muhammad Azzazy 900202821

---



# TABLE OF CONTENTS

01

## Roles

Roles of every team member

02

## Ranges Chosen

Descriptions priority range chosen and why

03

## Priority Semantics

Min value , max value , and default value

04

## Inputs Verification

How the inputs to the system calls and application were verified

05

## Test Cases

For system calls and application

06

## Pseudocode

Descriptions and pseudocode for the two scheduling algorithms

# TABLE OF CONTENTS

07

## Decay Factor

Details on the decay factor and how frequent priorities decay

08

## Scheduling Test Cases

Test cases for the scheduling algorithms

10

## XV6 Edited Files

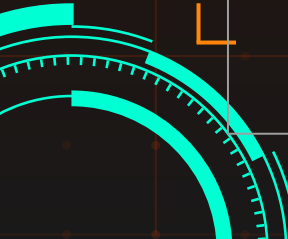
Test cases for the scheduling algorithms XV6 files that were modified or added and reasons for adding or modifying them



# Roles

---

Roles of every team member



Name	Role
Muhammad Azzazy	<ul style="list-style-type: none"><li>- Implemented aging by modifying the yield() function</li><li>- Decided on the decay factor</li><li>- Tested both scheduling algorithms</li><li>- Chose the ranges for priority including the default value</li></ul>
Mariam Dahab	<ul style="list-style-type: none"><li>- Added priority attributed</li><li>- Created set_priority and printptable system calls</li><li>- Created user programs to test system calls</li></ul>
Jana Saleh	<ul style="list-style-type: none"><li>- Implemented the scheduler priority algorithm (in scheduler function in proc.c)</li></ul>



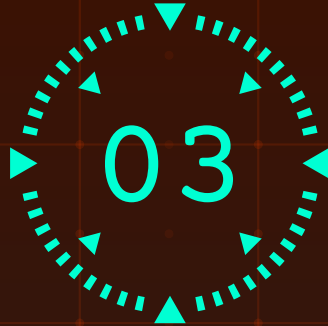
# Priority Ranges

---

Ranges chosen for priority and why

# Priority Ranges

- The maximum value for priority is 5.
- The minimum value for priority is 0.
- A smaller range of priority values can simplify the priority management and make it easier to understand.
- Developing a system with specific requirements or constraints requires a smaller range of priority values.



# Priority Semantics

---

Min value, max value, default value



# Priority Semantics

- The maximum value is enforced when using the `set_priority()` function. This was done by checking if the priority is less than 6. If it is, then another condition follows.
- The minimum value is enforced when using the `set_priority()` function. This was done by checking if the priority is greater than -1. If the priority is greater than -1, then assign the current priority value to the variable holding the old priority value and the new priority value to the priority value of the process.
- The default value is enforced by setting the priority of the process to 2 within the function `fork()`. This is done after allocating resources to the new child process. An if condition is used to check whether this is sh. If it is sh, then its priority is left unchanged.



# Inputs Verification

---

System calls and application verification of inputs

# Inputs Verification

System Call / Application	Verification
<code>setpr(pid,priority)</code>	The application checks that the new priority value is within the range before calling the system call function
<code>set_priority()</code>	The system call checks the validity of the pid and notifies user if id is invalid



# Test Cases

---

Print Process Table and Set Priority system  
calls' test cases

# Print Process Table - ppt()

```
1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4 #include "fcntl.h"
5
6 int main(void){
7
8     printtable();
9     int pid=fork();
10    printf(1,"-----\n");
11    printtable();
12    wait();
13    if(pid !=0) {
14        printf(1,"-----\n");
15        printtable();
16    }
17    exit();
18 }
```

```
init: starting sh
$ ppt
name o pid o status o priority
init o 1 o SLEEPING o 0
sh o 2 o SLEEPING o 0
ppt o 3 o RUNNING o 0
-----
name o pid o status o priority
init o 1 o SLEEPING o 0
sh o 2 o SLEEPING o 0
ppt o 3 o RUNNING o 0
ppt o 4 o RUNNABLE o 10
-----
name o pid o status o priority
init o 1 o SLEEPING o 0
sh o 2 o SLEEPING o 0
ppt o 3 o SLEEPING o 0
ppt o 4 o RUNNING o 10
-----
name o pid o status o priority
init o 1 o SLEEPING o 0
sh o 2 o SLEEPING o 0
ppt o 3 o RUNNING o 0
```

# Set Priority - setpr(pid , priority)

```
1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4 #include "fcntl.h"
5
6 int
7 main(int argc, char *argv[])
8 {
9     int pid , priority;
10    if(argc < 3){
11        printf(1,"Invalid Input ! Please enter Pid and new priority value \n");
12        exit();
13    }
14    pid = atoi(argv[1]);
15    priority = atoi(argv[2]);
16
17    if(priority < 0 || priority>31)
18    {
19        printf(1,"Invalid Input ! Please enter priority value in range 0-31\n");
20        exit();
21    }
22
23    int old = set_priority(pid,priority);
24
25    printf(1,"Old Value is %d \n ",old);
26    exit();
27 }
```

name	pid	status	priority
init	1	SLEEPING	0
sh	2	SLEEPING	0
ppt	3	RUNNING	0

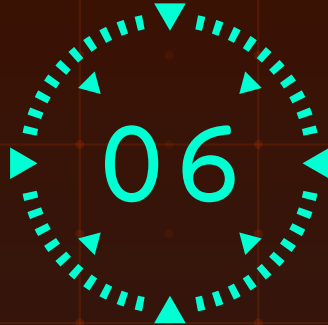
\$ setpr 2 3

Old Value is 0

\$ ppt

name	pid	status	priority
init	1	SLEEPING	0
sh	2	SLEEPING	3
ppt	6	RUNNING	0

-----



# PSEUDOCODES

---

# Priority scheduler algorithm

```
struct proc *p, *p1
struct cpu *c = mycpu()
c->proc = 0

for(;;)
    sti();
    struct proc *highpriority
    acquire(&ptable.lock)
    For ptable.proc to p < &ptable.proc[NPROC]
        if(p->state not RUNNABLE) continue
        highP = p
    For ptable.proc to p < &ptable.proc[NPROC]

    if((highpriority->priority > p1->priority) && (p1->state == RUNNABLE)){
        highP = p1
    else
        continue

        p = highpriority;
        c->proc = p;
        switchvm(p);
        p->state = RUNNING;
        swtch(&(c->scheduler), p->context);
        switchkvm();
        c->proc = 0;

    release(&ptable.lock);
```



# Priority Scheduling with Aging

Added these statement to the function yield()

```
decay_factor <= 1
```

```
set_priority(myproc()->pid, (myproc()->priority)-decay_factor);
```

This was done to apply the decay factor to the processes that are scheduled to run.



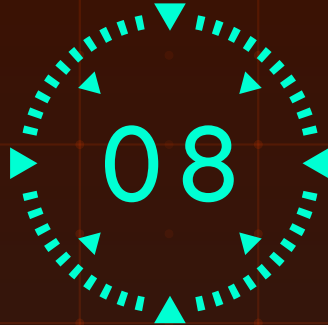
# Decay Factor

---

Details on the decay factor and how frequent priorities decay

# Decay Factor

- Decrementing the value of priority by 1 each time the process is scheduled to run is best because it is similar to what was suggested by Linux system engineers which involves halving the value of priority.
- However, since there is a short range of values for priority, namely from 0 to 5, we need to decrement the priority value instead of halving it.



# Testing Scheduling

---

Test cases for the scheduling algorithms with  
screenshots

# Priority Scheduling Only

```
$ sched 4
Testing priority scheduler...
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  2
sched 4    RUNNING  1
sched 5    RUNNABLE  2
sched 6    RUNNABLE  2
sched 7    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    RUNNABLE  2
sched 4    ZOMBIE    1
sched 5    RUNNING   2
sched 6    RUNNABLE  2
sched 7    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    RUNNABLE  2
sched 4    ZOMBIE    1
sched 5    RUNNABLE  2
sched 6    RUNNING   3
sched 7    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    RUNNABLE  2
sched 4    ZOMBIE    1
sched 5    RUNNABLE  2
sched 6    ZOMBIE    3
sched 7    RUNNING   4
Child process with pid 4 exited
Child process with pid 6 exited
Child process with pid 5 exited
Child process with pid 7 exited
Priority scheduler test complete
```

# Priority Scheduling Only

```
$ sched 5
Testing priority scheduler...
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    RUNNABLE  2
sched 4    RUNNING  1
      5    EMBRYO   2
Child process with pid 4 exited
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  2
sched 5    RUNNING  2
sched 6    RUNNABLE  2
sched 7    RUNNABLE  2
sched 8    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  2
sched 5    RUNNABLE  2
sched 6    RUNNING  3
sched 7    RUNNABLE  2
sched 8    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  2
sched 5    RUNNABLE  2
sched 6    RUNNABLE  3
sched 7    RUNNING  4
sched 8    RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    RUNNABLE  2
sched 5    RUNNABLE  2
sched 6    RUNNABLE  3
sched 7    ZOMBIE   4
sched 8    RUNNING  5
Child process with pid 7 exited
Child process with pid 5 exited
Child process with pid 6 exited
Child process with pid 8 exited
Priority scheduler test complete
```

# Priority Scheduling with Aging

```
$ sched 4
Testing priority scheduler...
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 10   SLEEPING  0
sched 11   RUNNING  1
sched 12   RUNNABLE  2
sched 13   RUNNABLE  2
sched 14   RUNNABLE  2
Child process with pid 11 exited
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 10   SLEEPING  0
sched 12   RUNNING  2
sched 13   RUNNABLE  2
sched 14   RUNNABLE  2
Child process with pid 12 exited
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 10   SLEEPING  0
sched 13   RUNNING  3
sched 14   RUNNABLE  2
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 10   SLEEPING  0
sched 13   RUNNABLE  2
sched 14   RUNNING  4
Child process with pid 14 exited
Child process with pid 13 exited
Priority scheduler test complete
```



# Priority Scheduling with Aging

Testing priority scheduler...

```
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  1
sched 4    RUNNING  1
sched 5    RUNNABLE  2
sched 6    RUNNABLE  2
sched 7    RUNNABLE  2
sched 8    RUNNABLE  2
```

Child process with pid 4 exited

```
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  1
sched 5    RUNNING  2
sched 6    RUNNABLE  2
sched 7    RUNNABLE  2
sched 8    RUNNABLE  2
```

Child process with pid 5 exited

```
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  1
sched 6    RUNNING  3
sched 7    RUNNABLE  2
sched 8    RUNNABLE  2
```

```
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  1
sched 6    RUNNABLE  2
sched 7    RUNNING  4
sched 8    RUNNABLE  2
```

Child process with pid 7 exited

Child process with pid 6 exited

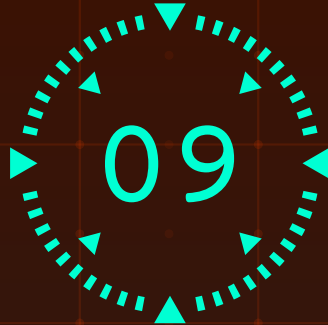
```
=====
name  pid  status  priority
init  1    SLEEPING  0
sh    2    SLEEPING  0
sched 3    SLEEPING  0
sched 8    RUNNING  5
```

Child process with pid 8 exited

Priority scheduler test complete

~





# Modified xv6 Files

---

Xv6 files that were modified or added and reasons  
for adding or modifying them

Task	File	Reason
Add priority attribute	proc.h	Add attribute to process struct
	proc.c	Set initial value to parent process, add decay factor to yield and applied it to processes, modified fork to set priority of child processes to default value of 2
	exec.c	Set initial value to child process
Set_priority and printptable system calls	syscall.h	Assign system call number
	user.h	Define function called from user program
	defs.h	Add Function definition
	proc.c	Add function implementation
	sysproc.c	Implement system call function
	usys.s	Interface for user program to access system call
	syscall.c	Add system call pointer and prototype
	Makefile	Add the new user programs
setpr() and ppt() user programs	setpr.c (created)	User program code to test set_priority system call
	printptable.c (created)	User program code to test printptable system call

# THANKS

Do you have any questions?

[janasaleh@aucegypt.edu](mailto:janasaleh@aucegypt.edu)

[mariamhdahab@aucegypt.edu](mailto:mariamhdahab@aucegypt.edu)

[muhammad-azzazy@aucegypt.edu](mailto:muhammad-azzazy@aucegypt.edu)



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**

Please keep this slide for attribution