

SER 334 A Session

SI Session

Monday, January 29th 2024

7:00 pm - 8:00 pm MST

Agenda



Memory

Structure

Interprocess Communication

Process Tracing

Module 5 Samples

SI Session Expectations

Thanks for coming to the **SER 334** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
 - tutoring.asu.edu
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

Interact with us:

Zoom Features



Zoom Chat

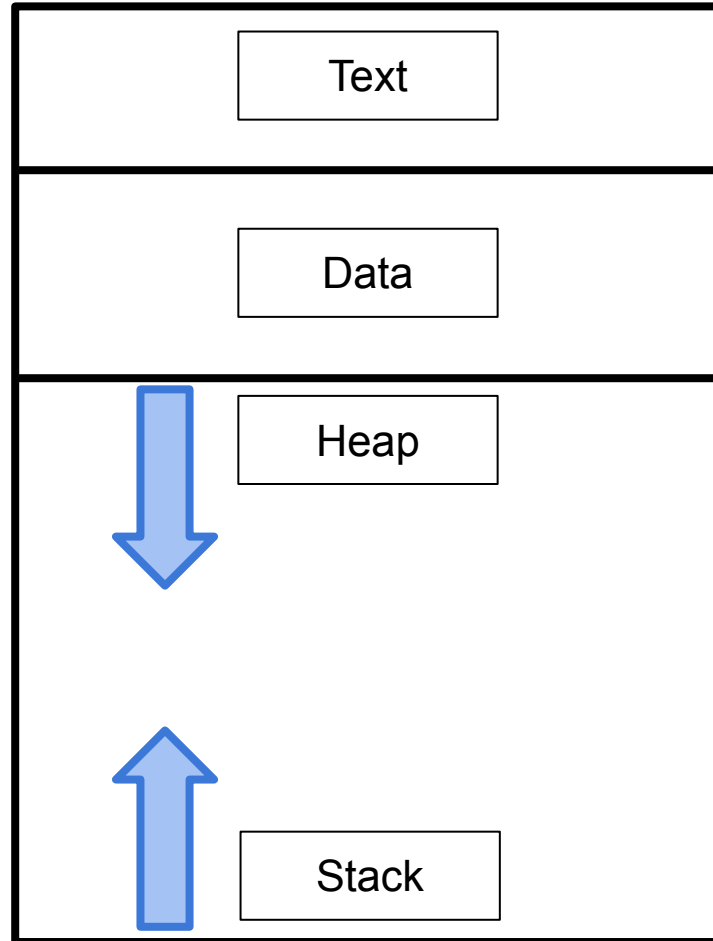
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 334

Memory

Which one holds
dynamically
allocated memory?

Memory

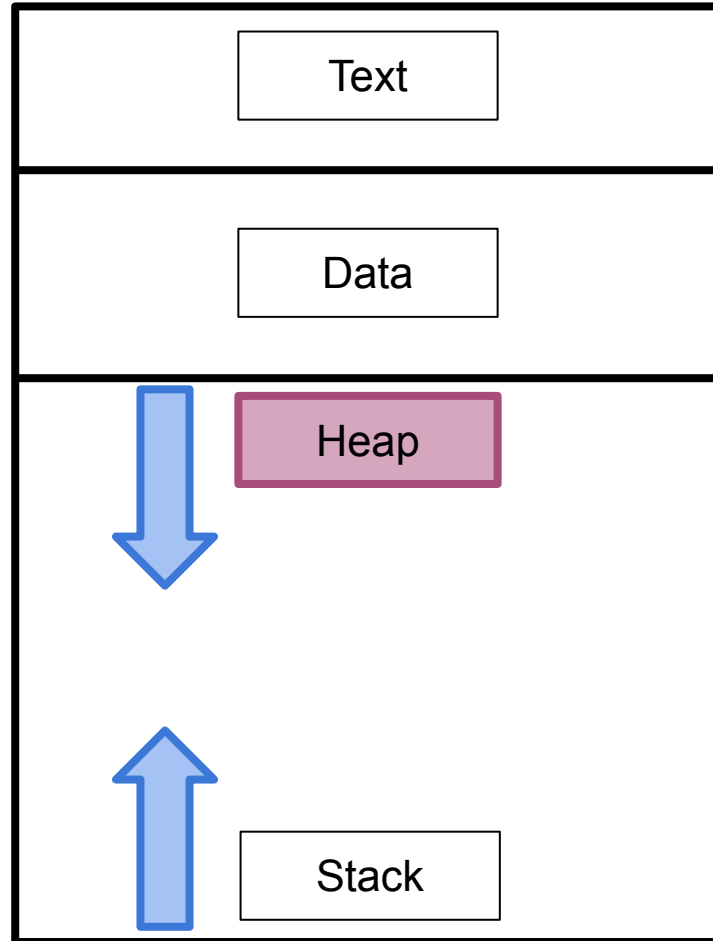


SER 334

Memory

Which one holds
dynamically
allocated memory?

Memory

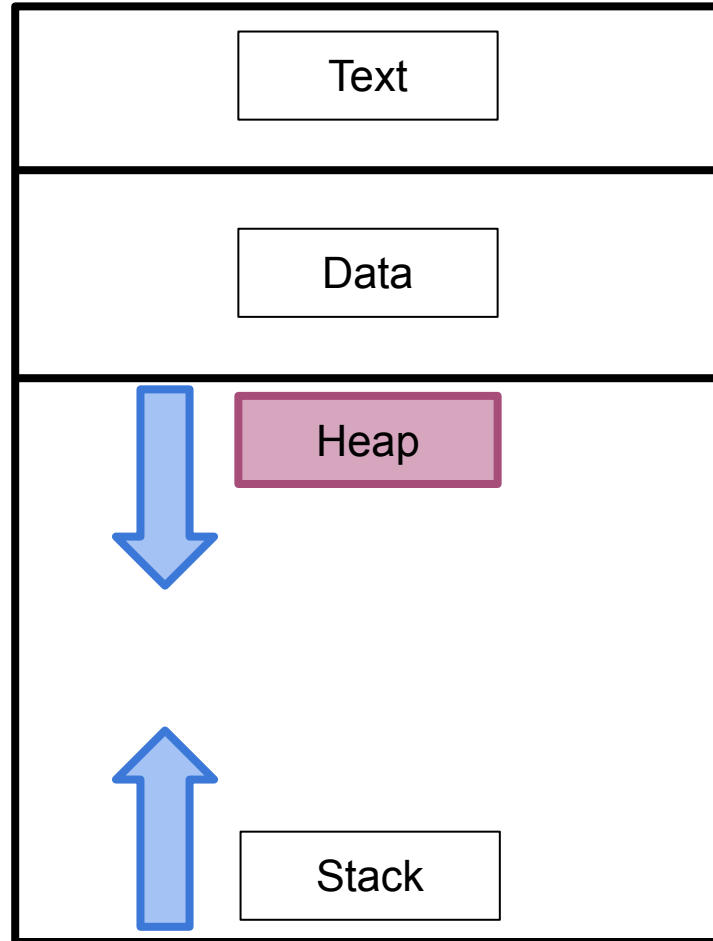


SER 334

Memory

Which one holds
our program code?

Memory

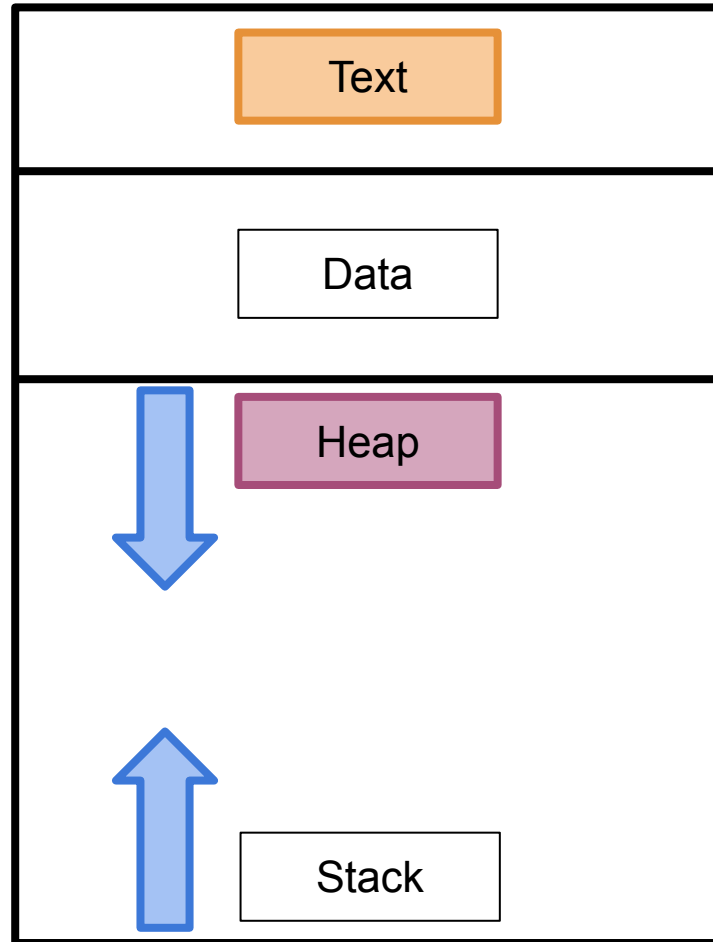


SER 334

Memory

Which one holds
our program code?

Memory

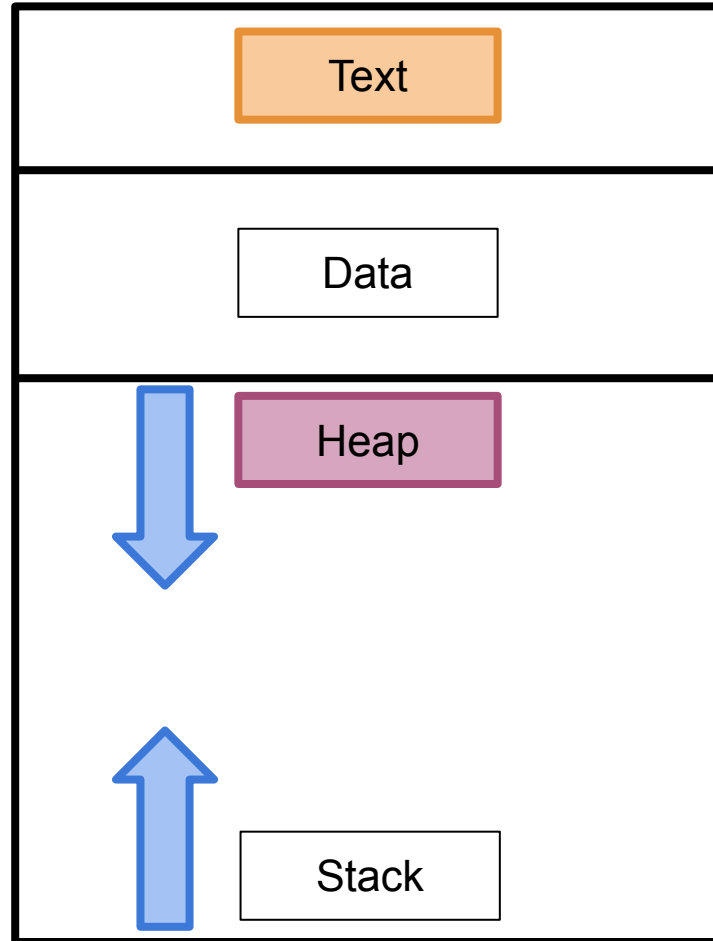


SER 334

Memory

Which one holds
global and static
variables and
objects?

Memory

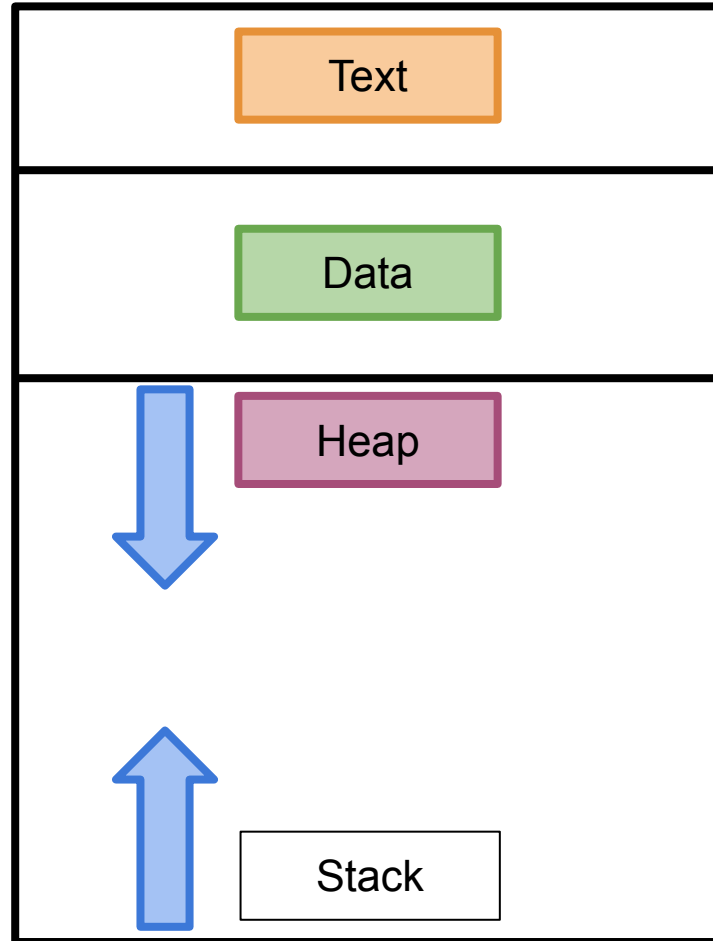


SER 334

Memory

Which one holds
global and static
variables and
objects?

Memory

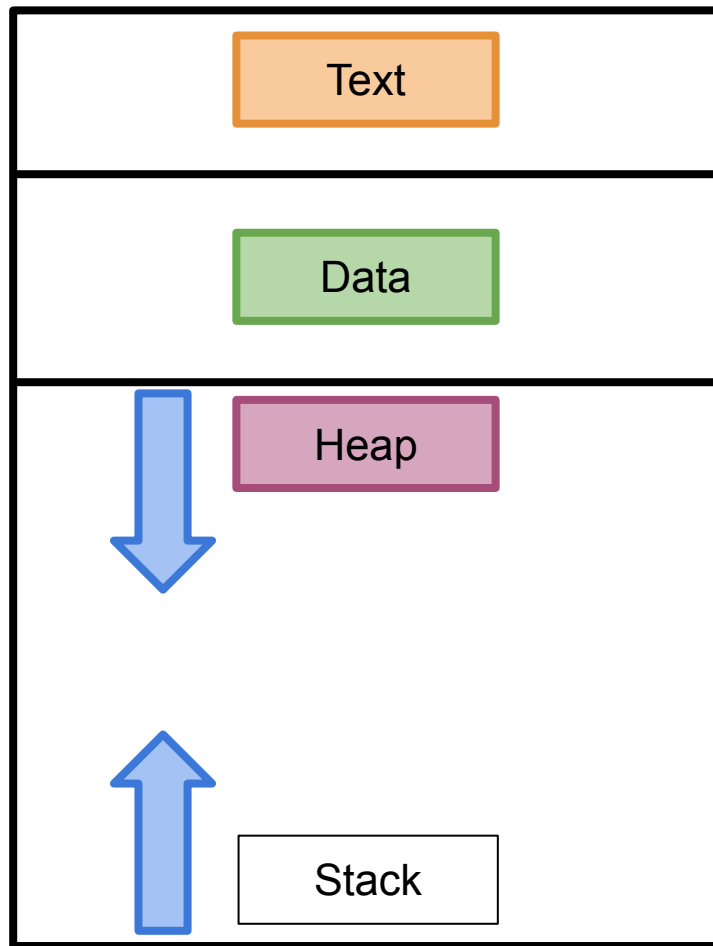


SER 334

Memory

Which one holds
temporary data?

Memory

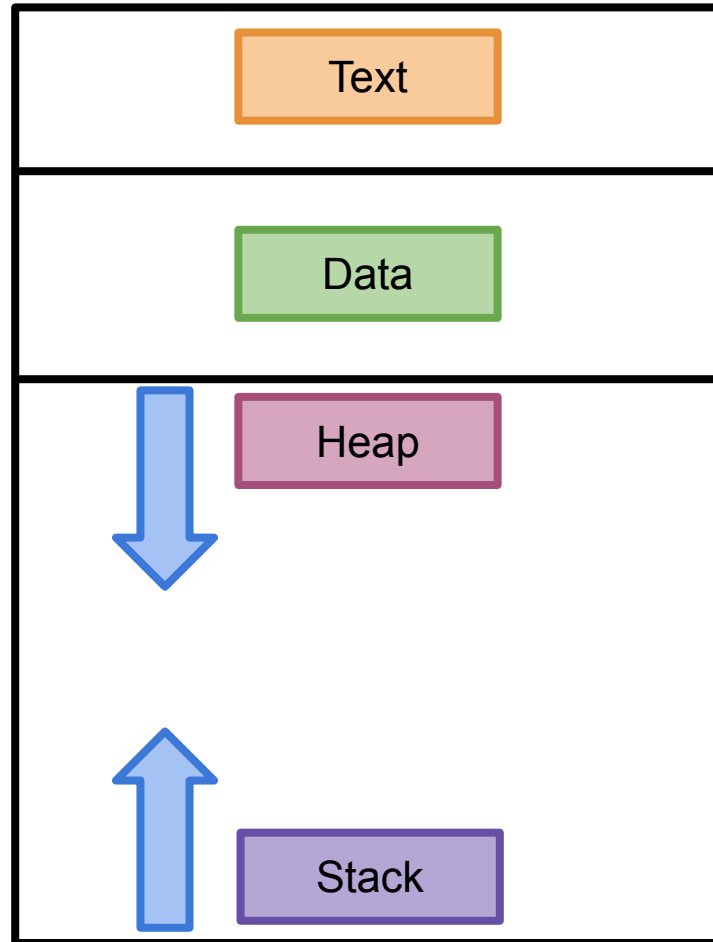


SER 334

Memory

Which one holds
temporary data?

Memory

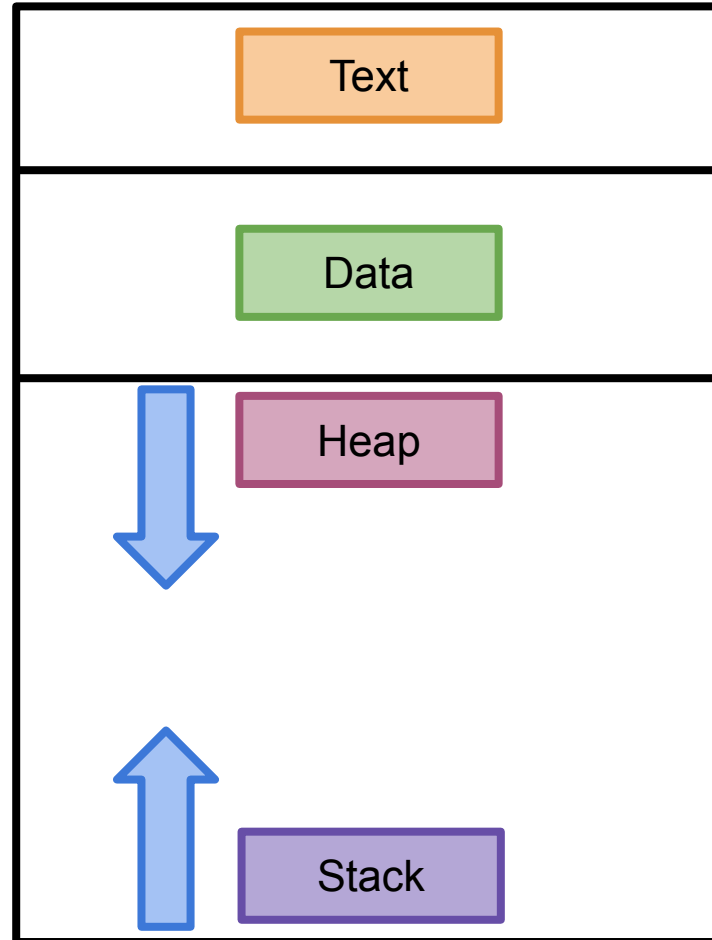


Memory

SER 334

Memory

1. [Acuña] There are four general regions to a process in memory: stack, heap, data, and text. **List and explain which of these must be separated for each program running and which might be combined globally.** (Don't worry about security or programs misbehaving.) [2 points]



SER 334

Memory

Shared Memory



vs.

Message Passing



Module 5 Sample

SER 334 Processes

pid1

pid2

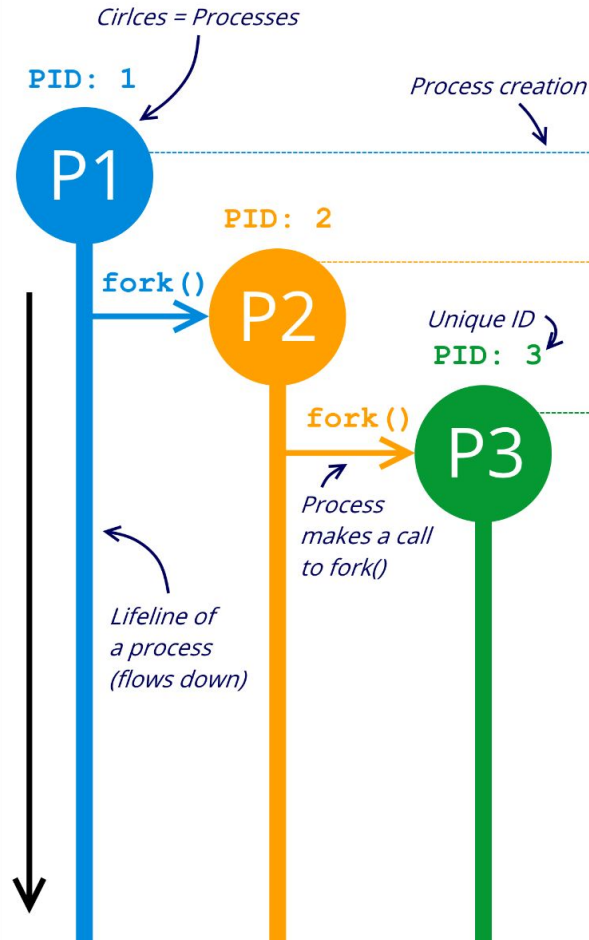
pid1

pid2

pid1

pid2

Output:



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

Lines of code touched per each process

SER 334

Processes

pid1 2

pid2

Why 2?

pid1

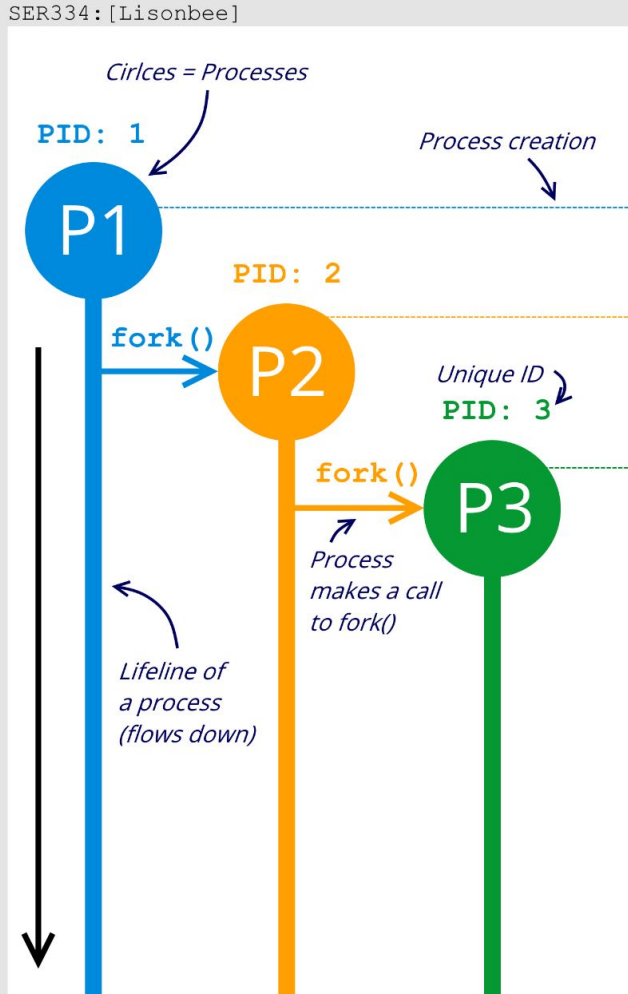
pid2

Why 0?

pid1

pid2

Output:



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

Lines of code touched per each process

SER 334

Processes

pid1 2

pid2

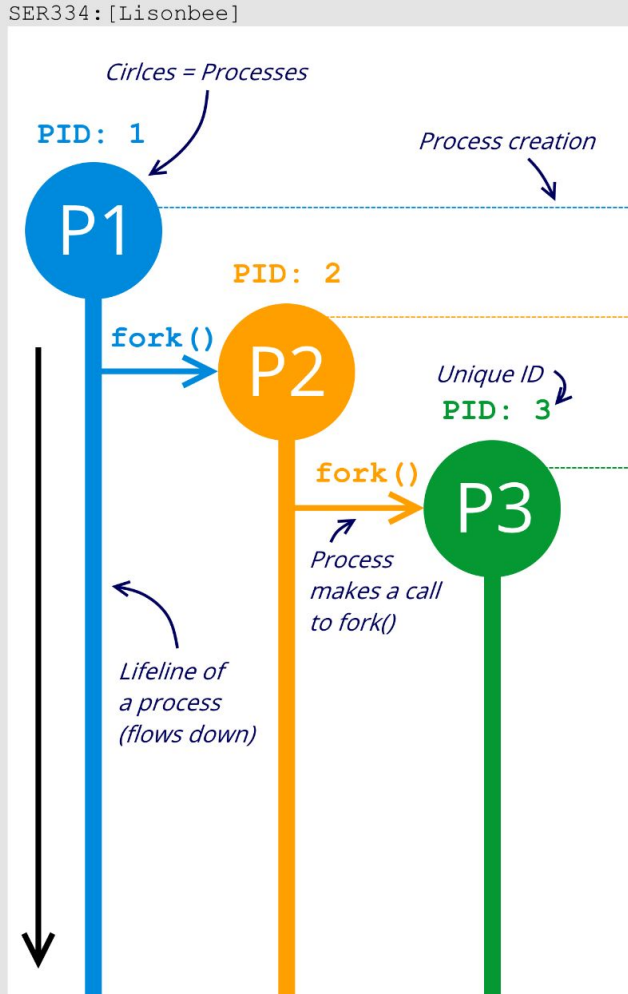
pid1 0

pid2 0

pid1

pid2

Output:



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

Lines of code touched per each process

SER 334

Processes

pid1 2

pid2

pid1 0

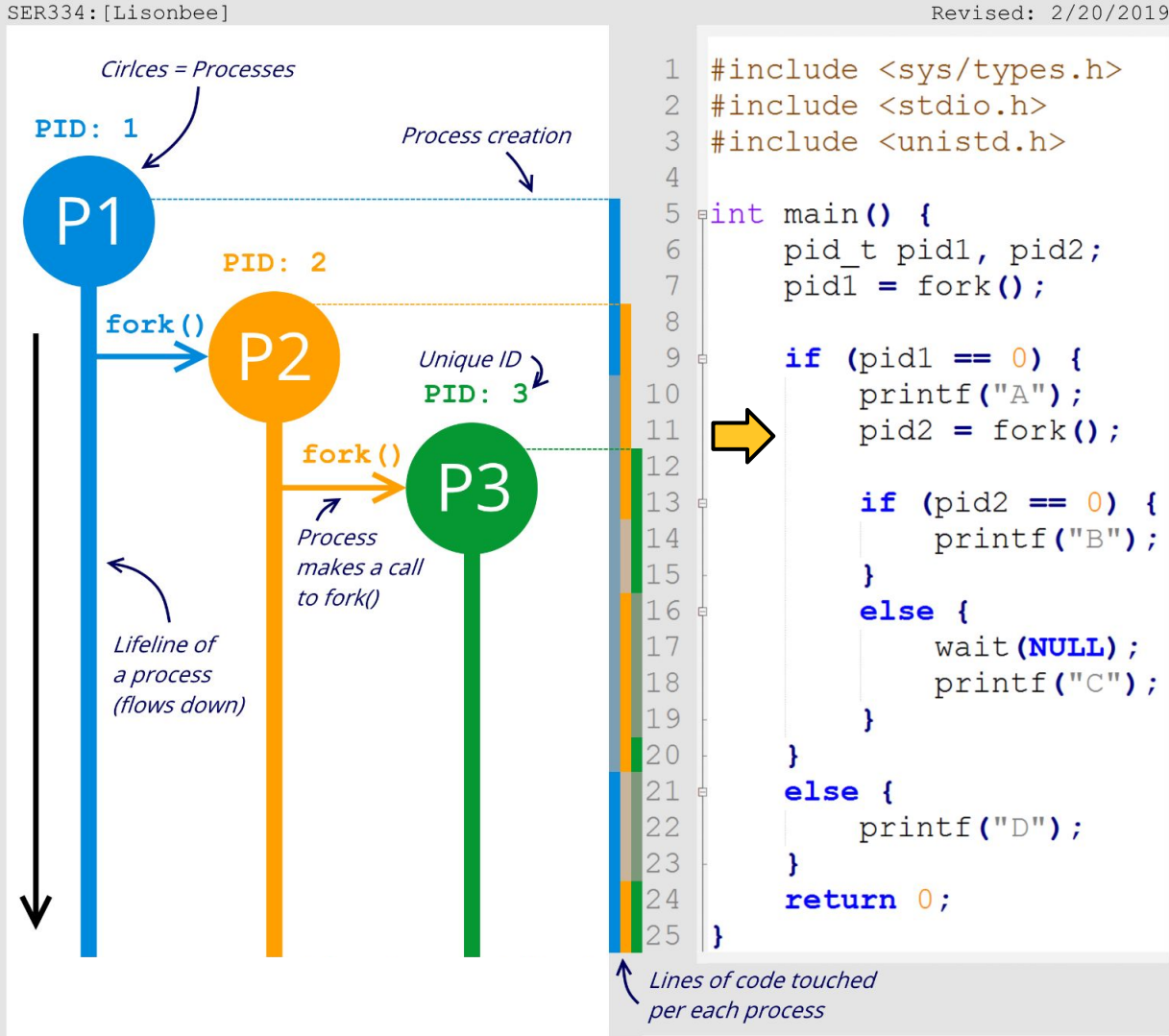
pid2 0

pid1

pid2

Output:

A



SER 334

Processes

pid1 2

pid2

pid1 0

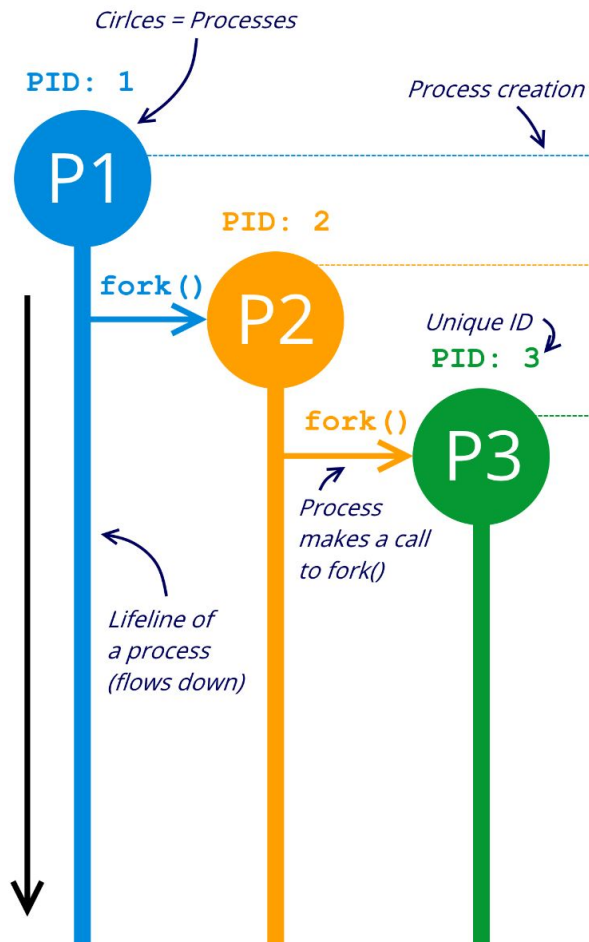
pid2 3

pid1 0

pid2 0

Output:

A B



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

*Lines of code touched
per each process*

SER 334**Processes**

pid1 2

pid2

pid1 0

pid2 3

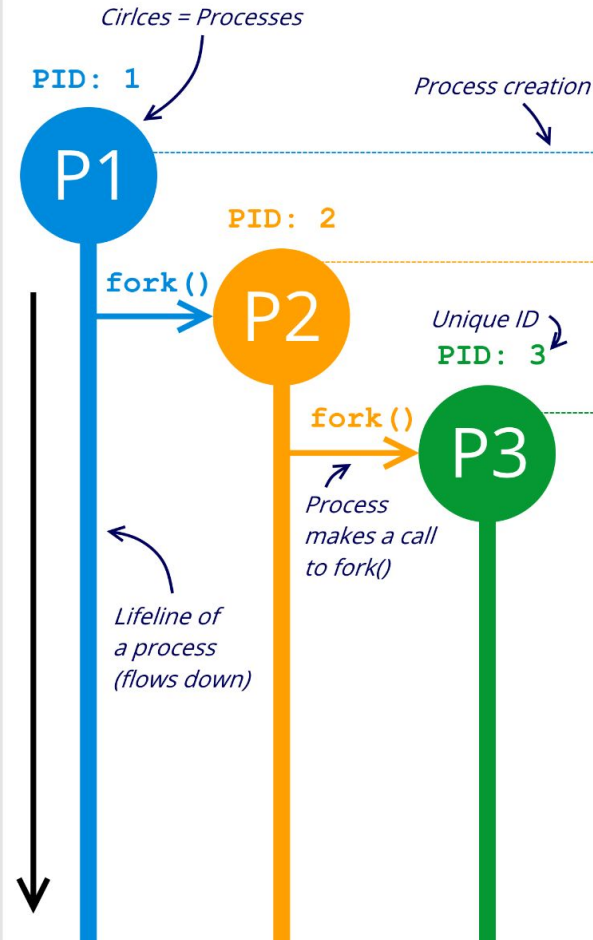
pid1 0

pid2 0

Which process
printed "B"?

Output:

A B



```

1  #include <sys/types.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  int main() {
6      pid_t pid1, pid2;
7      pid1 = fork();
8
9      if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

*Lines of code touched
per each process*

SER 334

Processes

pid1 2

pid2

pid1 0

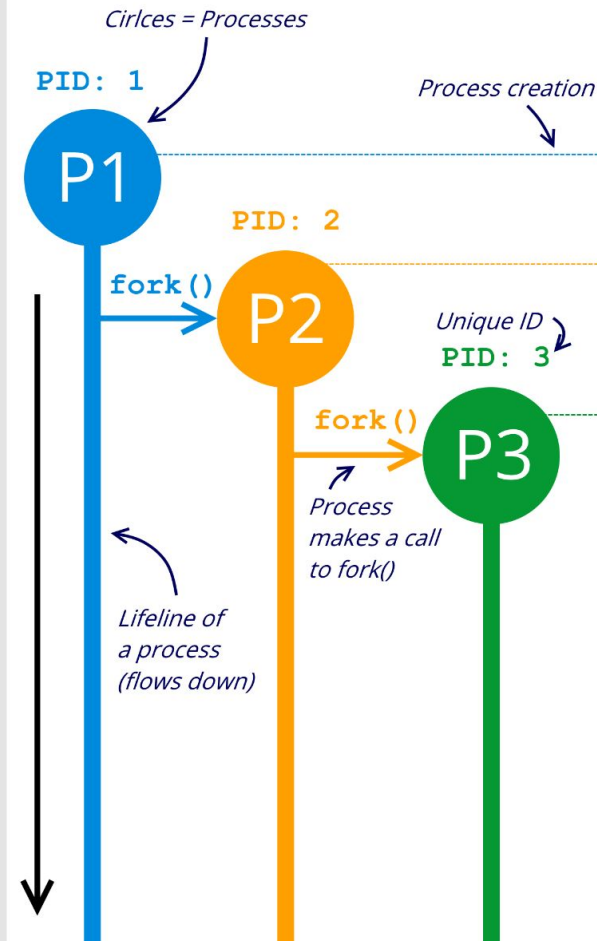
pid2 3

pid1 0

pid2 0

Output:

A B



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             ? wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

Lines of code touched per each process

SER 334

Processes

pid1 2

pid2

pid1 0

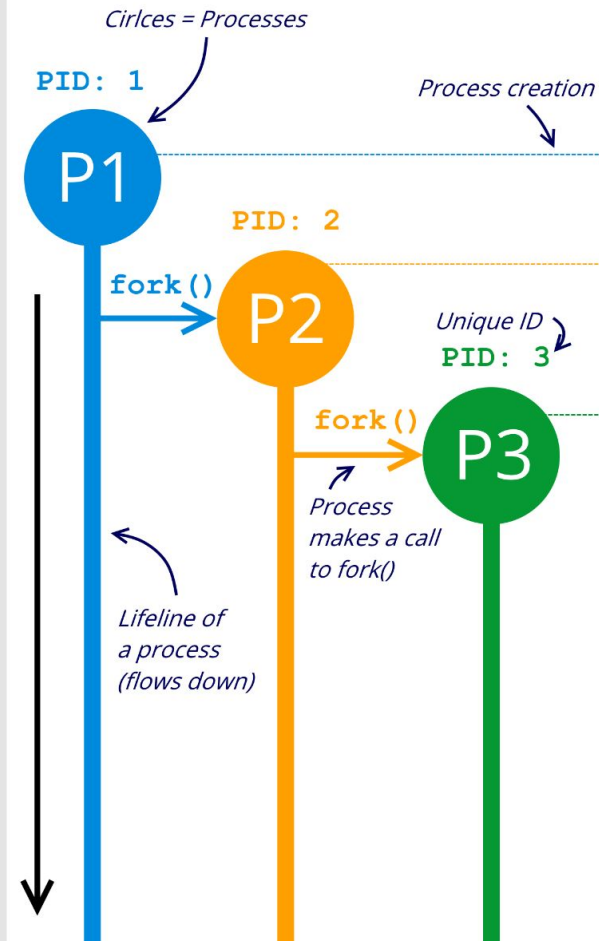
pid2 3

pid1 0

pid2 0

Output:

A B C



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

Lines of code touched per each process

SER 334

Processes

pid1 2

pid2

pid1 0

pid2 3

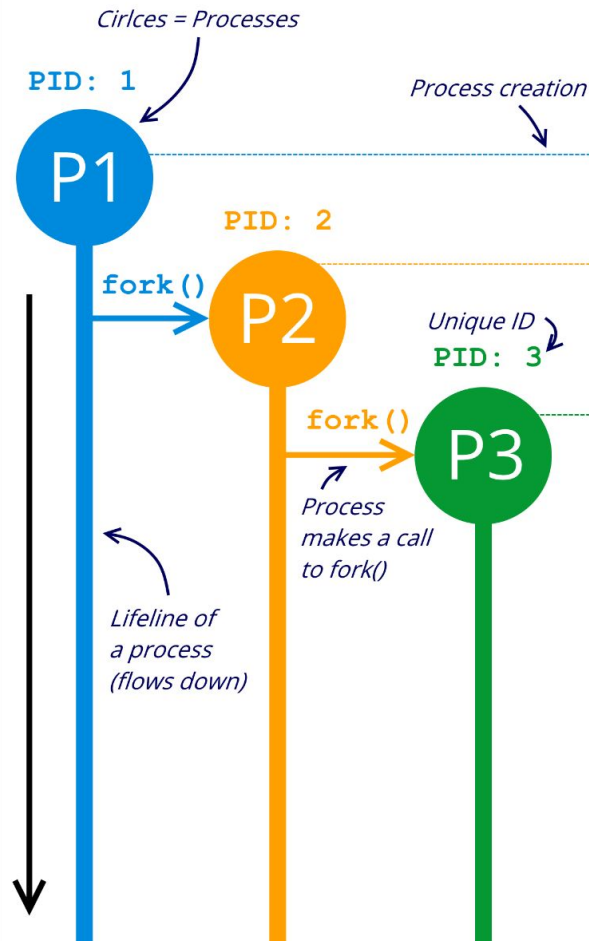
pid1 0

pid2 0

Which process
printed "D"?

Output:

A B C D



```

1 #include <sys/types.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     pid_t pid1, pid2;
7     pid1 = fork();
8
9     if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }

```

*Lines of code touched
per each process*

SER 334**Processes**

Is that the only solution?

pid1 2

pid2

pid1 0

pid2 3

pid1 0

pid2 0

Output:

A B C D

fork()

P2

Unique ID
PID: 3

fork()

Process
makes a call
to fork()

P3

Lifeline of
a process
(flows down)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
```

```
int main() {
    pid_t pid1, pid2;
    pid1 = fork();

    if (pid1 == 0) {
        printf("A");
        pid2 = fork();

        if (pid2 == 0) {
            printf("B");
        }
        else {
            wait(NULL);
            printf("C");
        }
    }
    else {
        printf("D");
    }
    return 0;
}
```

Lines of code touched
per each process

SER 334

Processes

Is that the only solution?

pid1 2

pid2

pid1 0

pid2 3

pid1 0

pid2 0

Output:

A B C D

fork()

P2

Unique ID
PID: 3

fork()

Process
makes a call
to fork()

P3

Lifeline of
a process
(flows down)

A

B C

D

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
```

```
int main() {
    pid_t pid1, pid2;
    pid1 = fork();

    if (pid1 == 0) {
        printf("A");
        pid2 = fork();

        if (pid2 == 0) {
            printf("B");
        }
        else {
            wait(NULL);
            printf("C");
        }
    }
    else {
        printf("D");
    }
    return 0;
}
```

Lines of code touched
per each process

SER 334**Processes**

pid1 2

pid2

pid1 0

pid2 3

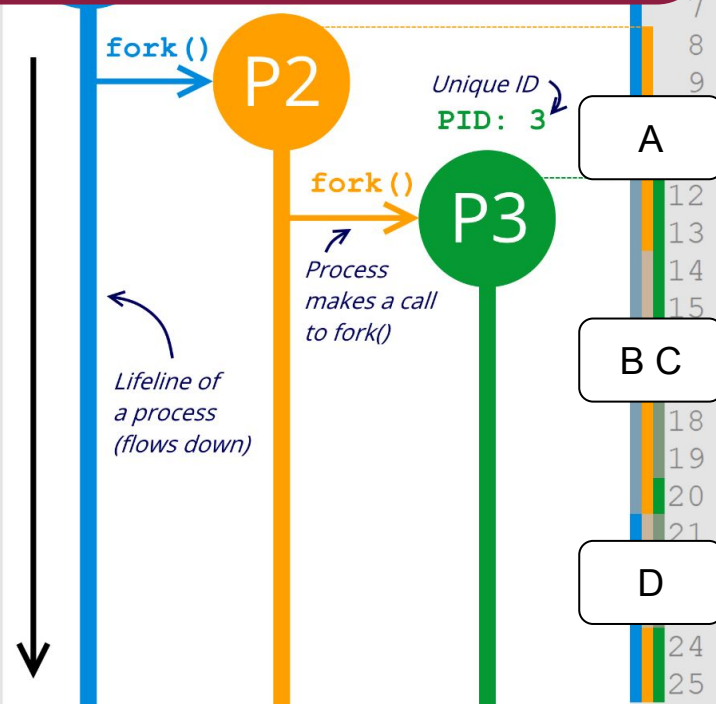
pid1 0

pid2 0

Output:

ABCD
ADBC
DABC
ABDC

Is that the only solution?



```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
```

```
int main() {
    pid_t pid1, pid2;
    pid1 = fork();

    if (pid1 == 0) {
        printf("A");
        pid2 = fork();

        if (pid2 == 0) {
            printf("B");
        }
        else {
            wait(NULL);
            printf("C");
        }
    }
    else {
        printf("D");
    }
    return 0;
}
```

Lines of code touched
per each process

SER 334
Module 5 Sample

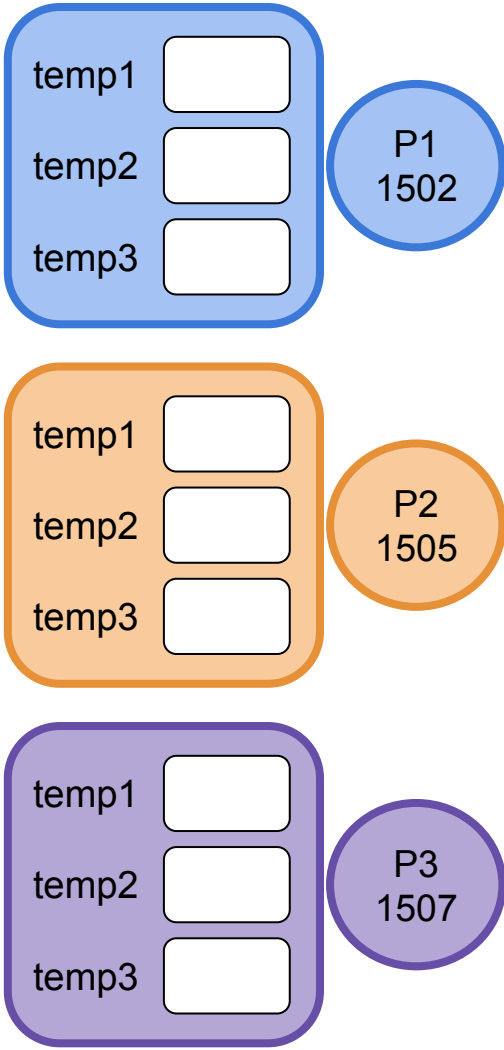
4. [Lisonbee] Trace the program below, identify the values of the pids at lines A, B, C, D, E, and F. (Assume that the actual pid of Process 1 is 1502, Process 2 is 1505, and Process 3 is 1507. Also assume that fork will always succeed.) [4 points]

A	
B	
C	
D	
E	
F	

```
int main() {
    pid_t temp1, temp2, temp3;
    temp1 = fork();

    if (temp1 < 0) { /* Error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (temp1 == 0) { /* Process 2 */
        temp2 = fork();

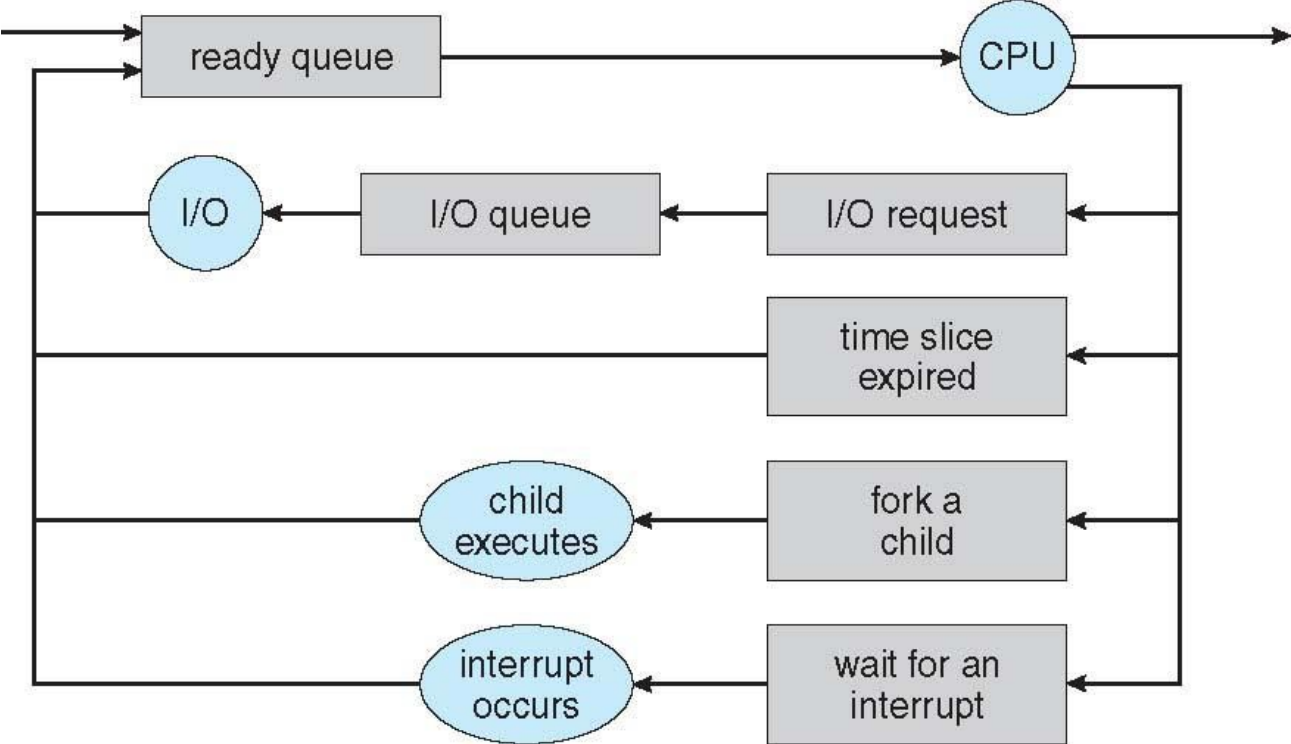
        if (temp2 < 0) { /* Error occurred */
            fprintf(stderr, "Fork Failed");
            return 1;
        }
        else if (temp2 == 0) { /* Process 3 */
            temp3 = getpid();
            printf("temp2 = %d", temp2); /* A */
            printf("temp3 = %d", temp3); /* B */
        }
        else { /* Process 2 */
            temp3 = getpid();
            printf("temp2 = %d", temp2); /* C */
            printf("temp3 = %d", temp3); /* D */
            wait(NULL);
        }
    }
    else { /* Process 1 */
        temp2 = getpid();
        printf("temp1 = %d", temp1); /* E */
        printf("temp2 = %d", temp2); /* F */
        wait(NULL);
    }
    return 0;
}
```



SER 334

Enter topic here

3. [Bahremand] Explain why the queue diagram in slide 9 has a continuous cycle that flows between the listed queues and resources? Is a cycle necessary? [2 points]



SER 334

Enter topic here

6. [Lisonbee] Consider a system where two processes (a producer and a consumer) use a message-passing system to communicate, and each process does work at different rates. The producer can produce (perform work) at any rate, but the consumer has to wait for the producer to complete its task before moving on. Based on this system's needs, explain whether a synchronous or asynchronous communication system would be a better choice and why. [2 points]

Synchronous

Asynchronous

SER 334

Scratch Space

Upcoming Events

SI Sessions:

- Thursday, February 1st at 7:00 pm MST
- Sunday, February 4th at 7:00 pm MST
- Monday, February 5th at 7:00 pm MST

Review Sessions:

- Exam 2 Review: TBD
- Exam 3 Review: TBD

Questions?

Survey:

<http://bit.ly/ASN2324>



More Questions?

Check out our other resources!

tutoring.asu.edu



Academic Support

Academic Support Network (ASN) provides a variety of free services in-person and online to help currently enrolled ASU students succeed academically.

Services



Subject Area Tutoring

Need in-person or online help with math, science, business, or engineering courses? Just hop into our Zoom room or drop into a center for small group tutoring. We'll take it from there.

[Need help using Zoom?](#)

[View the tutoring schedule](#)

[View digital resources](#)

Go to Zoom



Writing Tutoring

Need help with undergraduate or graduate writing assignments? Schedule an in-person or online appointment, access your appointment link, or wait in our drop-in queue.

[Access your appointment link](#)

[Access the drop-in queue](#)

Schedule Appointment



Online Study Hub

Join our online peer communities to connect with your fellow Sun Devils. Engage with our tools to search our bank of resources, videos, and previously asked questions. Or, ask our Tutorbot questions.

Now supporting courses in Math, Science, Business, Engineering, and Writing.

Online Study Hub

1-

Go to Zoom

2-

[Need help using Zoom?](#)

[View the tutoring schedule](#)

[View digital resources](#)






1. Click on 'Go to Zoom' to log onto our Online Tutoring Center.
2. Click on 'View the tutoring schedule' to see when tutors are available for specific courses.

More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub

 **Academic Support Network**

 [Services](#)  [Faculty and Staff Resources](#) [About Us](#) 

[University College](#)

Online Study Hub

Online peer communities for students and tutors, YouTube channels, and Tutorbots.



What are online peer communities?

Individual courses have an online peer community that allows you to connect with your peers to post and answer questions and to develop study groups.



How can tutoring center videos help?

Videos can help supplement the learning you're doing in and outside of class and include step-by-step methods for how to understand concepts.



How does the Tutorbot work?

You can ask the Tutorbot questions about course concepts and the Tutorbot will recommend additional resources and examples to help address your questions.

Select a subject

- Any -

[Apply](#)



Academic Support Network



[Services](#) 

[Faculty and Staff Resources](#)

[About Us](#) 

[University College](#)

Select a subject

- Any -

[Apply](#)

Business


ACC 231

Uses of Accounting Info I

 [Peer Community](#)

ACC 241

Uses of Accounting Info II

 [Peer Community](#)

CIS 105

Computer Applications and Information Technology

 [Peer Community](#)

Don't forget to check out the Online Study Hub for additional resources!

Additional Resources

- [Course Repo](#)
- [Course Discord](#)
- [BMP File Format \(Wiki\)](#)
- [Linux Kernel API](#)
- [Bootlin - Linux Cross Referencer](#)
- [Dining Philosophers Interactive](#)