



# Lab 2 More

---

Tips and DD Review



# Admin

---

- Lab 2 Design due next Monday!
  - Lab 2 Code + Questions due 10/28/24 (grace period and late days)
- Pset 2 due tomorrow 10/18/24
- Pset 3 Out!
  - Due 10/25/24

# Agenda

---

- Pipe Implementation Hints
- Exec Implementation Hints
- Lab 2 Design Doc Reviews

# Pipe Hints

---

# Pipe Impl Hints

- Remember, Pipe is a variant of the bounded buffer problem
  - producer = writer
  - consumer = reader



# Pipe Impl Hints

---

There are a lot of cases you will need to cover with your pipe design...

So let's discuss them!



# Pipe Impl Hints

---

- When should a writer wait?

# Pipe Impl Hints

---

- When should a writer wait?

When there is no room to write **and** still readers left



# Pipe Impl Hints

---

- When should a writer wait?

When there is no room to write **and** still readers left

- When should a reader wait?

# Pipe Impl Hints

---

- When should a writer wait?

When there is no room to write **and** still readers left

- When should a reader wait?

When there are no bytes to read **and** still writers left.

# Pipe Impl Hints

---

Say all writers are closed...

- What if there are sleeping readers? What should happen?

Say a new reader comes in...

- What should happen if there are active writers?
- What should happen if the writers are closed?
  - What if there is still data in the buffer?
  - What if there is no data in the buffer?

# Pipe Impl Hints

---

Are partial reads allowed? What about partial writes?

# Pipe Impl Hints

---

Are partial reads allowed? What about partial writes?

- Partial reads - YES
- Partial writes - NO

Ok... so how will you ensure that writes remain atomic?

# Pipe Impl Hints

---

In the slides for Lab 2 Part 2, we mentioned that part of the pipe metadata you need to track the waiting active writer...when and how do you use this information?

# Pipe Impl Hints

---

In the slides for Lab 2 Part 2, we mentioned that part of the pipe metadata you need to track the waiting active writer...when and how do you use this information?

- When a writer does not finish its write, we track it and block other writers!

# Exec Hints

---



# One More Look at `main()`

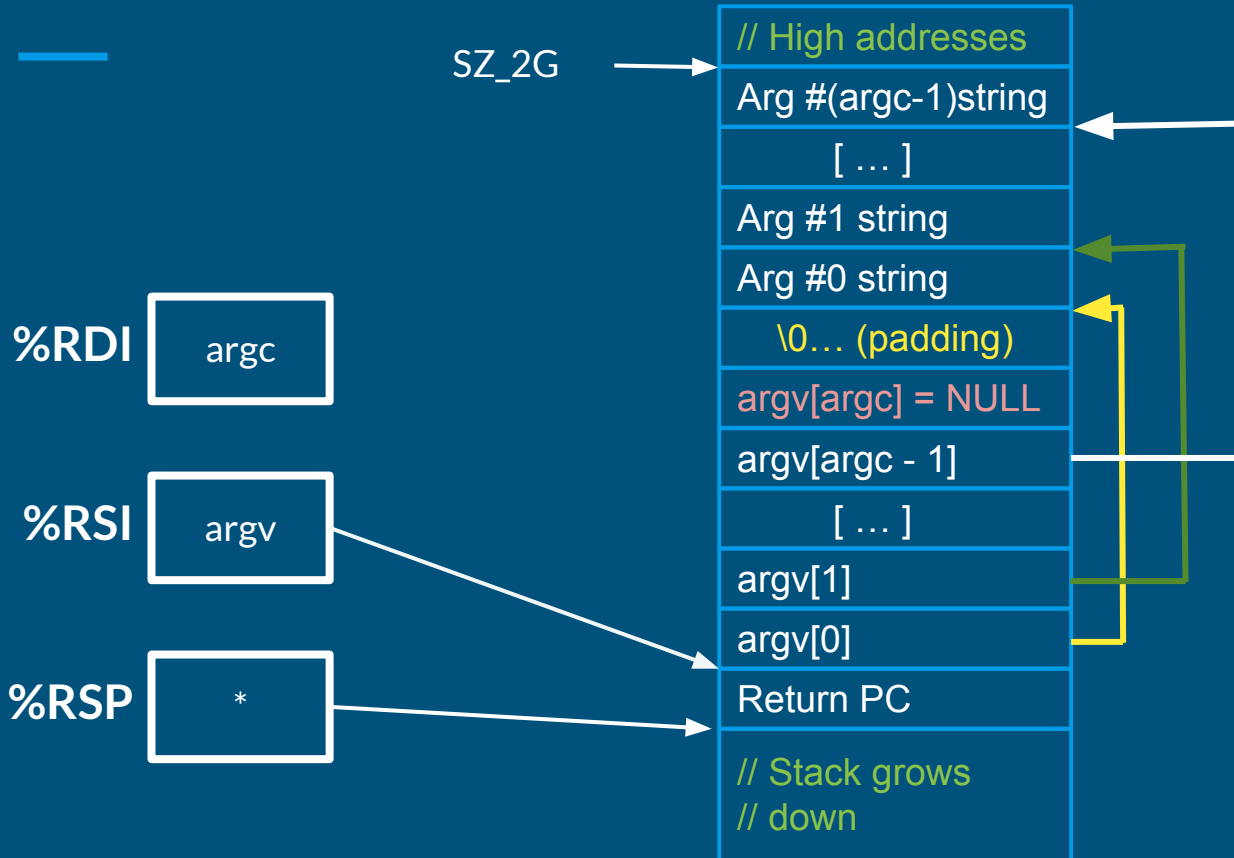
---

`exec` sets up the function arguments for `main`!

```
int main(int argc, char** argv)
```

- `argc`: The number of elements in `argv`
- `argv`: An array of strings representing program arguments
  - First is always the name of the program
  - `Argv[argc] = 0`

# One More Look at the Stack For User Process



- Since `argv` is an array of pointers, `%RSI` points to an array on the stack
- Since each element of `argv` is a `char*`, each element points to a string elsewhere on the stack
- **Why? Alignment**
- **Why NULL pointer? Convention**

# Exec Impl Hints

---

- Does the Return PC matter in xk?

# Exec Impl Hints

---

- Does the Return PC matter in xk?
  - Not really :)
  - The return pc is never used, since `main()` isn't called by anything. It doesn't matter what the value is, as long as it's 8 bytes.

# Exec Impl Hints

---

If you find yourself triple faulting when running the tests:

- Check when you install the new vspace
- Check when you free the old vspace

# More Lab 2 Part 2 Test Reminders

---

- Exec tests require a functioning pipe implementation!
- Just because the pipe tests pass now does not mean they will pass in lab3 and lab4 tests
  - Try to cover as many cases as you can with your pipe design (don't be lazy)
  - Write clean and easy-to-follow code when integrating the pipe into your File API logic

# Lab 2 Design Doc Review

---