POSIX I/O

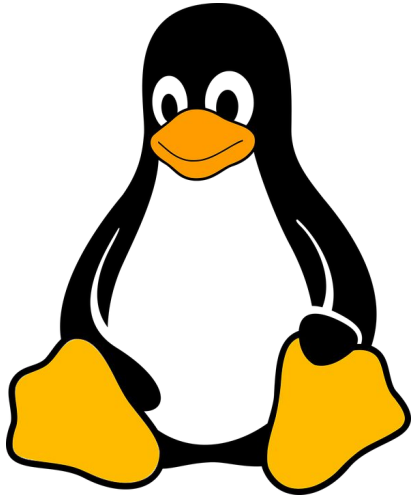
The fun stuff!

Review from Lecture

```
ssize_t read(int fd, void *buf, size_t count)
```

An error occurred	`result = -1` `errno = error`
Already at EOF	`result = 0`
Partial Read	`result < count`
Success!	`result == count`

New Scenario - Messy Roommate



- The Linux kernel is now your roommate
- There are N pieces of trash in the room
- There is a single trash can, `char bin[N]`
  - (For some reason, the trash goes in a particular order)
- You can tell your roommate to pick it up, but he/she is unreliable

New Scenario - Messy Roommate

`NumTrash pickup(roomNum, trashCan, Amount)`

"*I tried to start cleaning, but something came up*" (got hungry, had a midterm, room was locked, etc.)	`NumTrash == -1` `errno == excuse`
"*You told me to pick up trash, but the room was already clean*"	`NumTrash == 0`
"*I picked up some of it, but then I got distracted by my favorite show on Netflix*"	`NumTrash < Amount`
"*I did it! I picked up all the trash!*"	`NumTrash == Amount`

`NumTrash pickup(roomNum, trashCan, Amount)`

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`

How do we get the room clean?

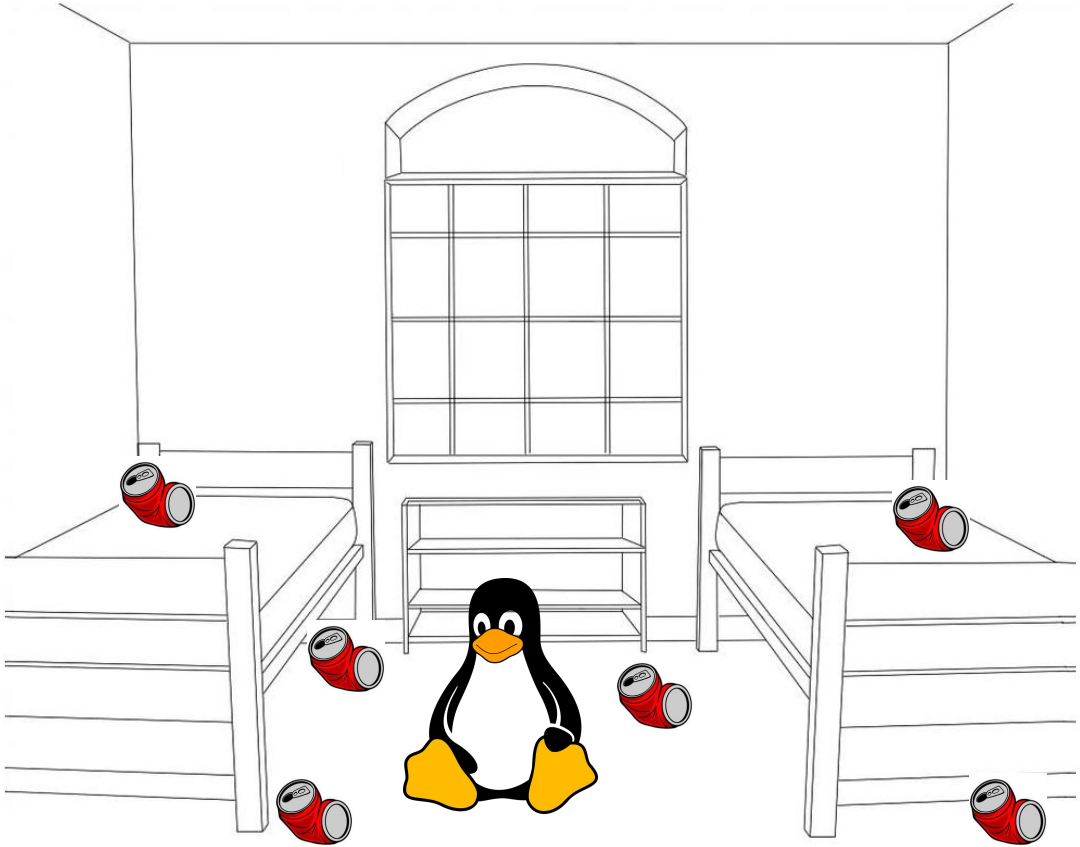
- Use a loop. What's the (high level) goal?
  - Pick up all N pieces of trash
- What if the roommate returns -1 with an excuse?
  - If it's a valid excuse, stop telling them to pick up trash
  - If it's not, start over at the top of the loop
- What if the room is already clean?
  - Stop telling the roommate to pick up trash
- What if the roommate only picked up some of it?
  - Record how much they picked up, and tell them to pick up the rest
- What if the roommate picked up everything you asked?
  - Our goal has been reached!

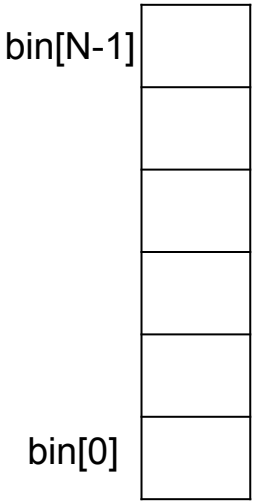
That's it!

How do we get the room clean?

```
NumTrash pickup(roomNum, trashCan, Amount)
```

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`



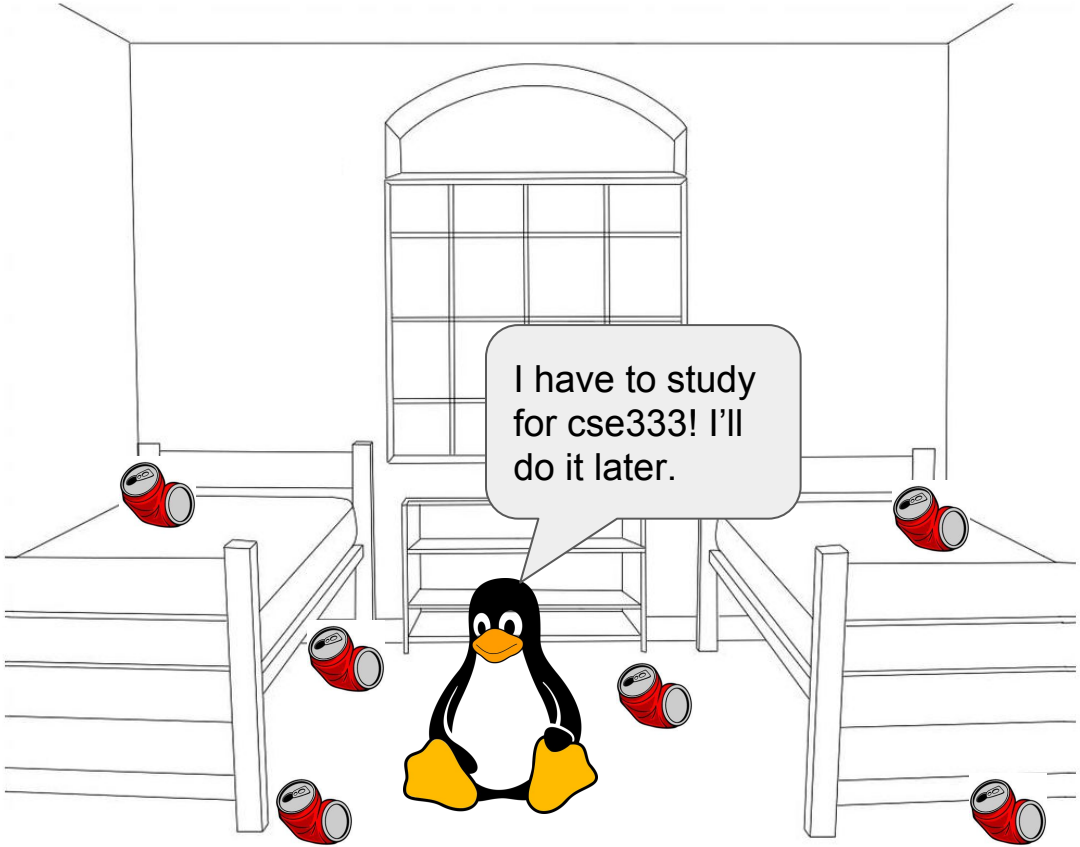


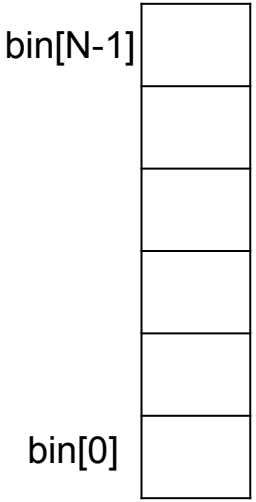
What do we do in the following scenarios?

How do we get the room clean?

```
NumTrash pickup(roomNum, trashCan, Amount)
```

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`



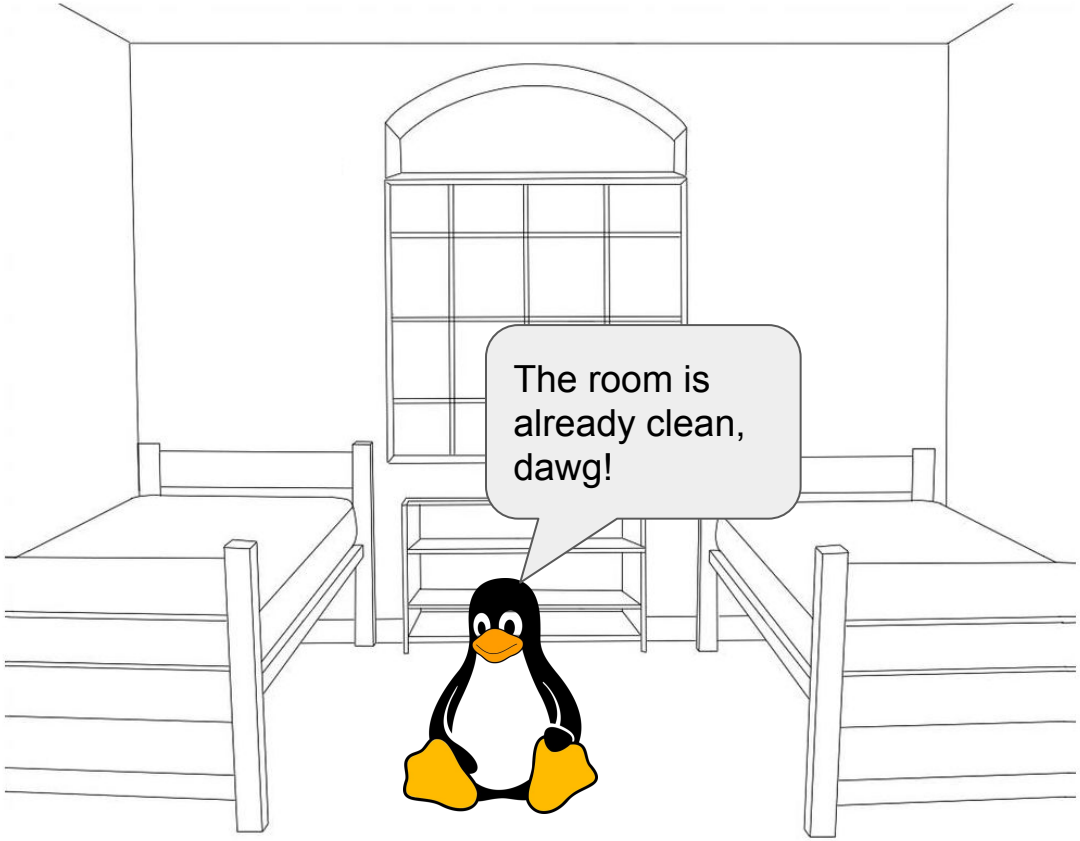


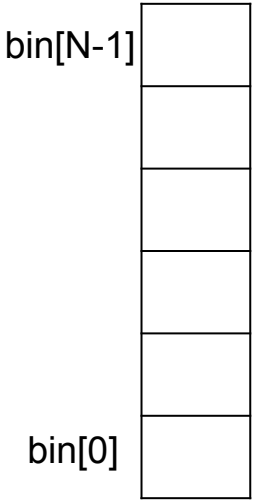
Decide if the excuse is reasonable, and either let it be or ask again.

How do we get the room clean?

```
NumTrash pickup(roomNum, trashCan, Amount)
```

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`



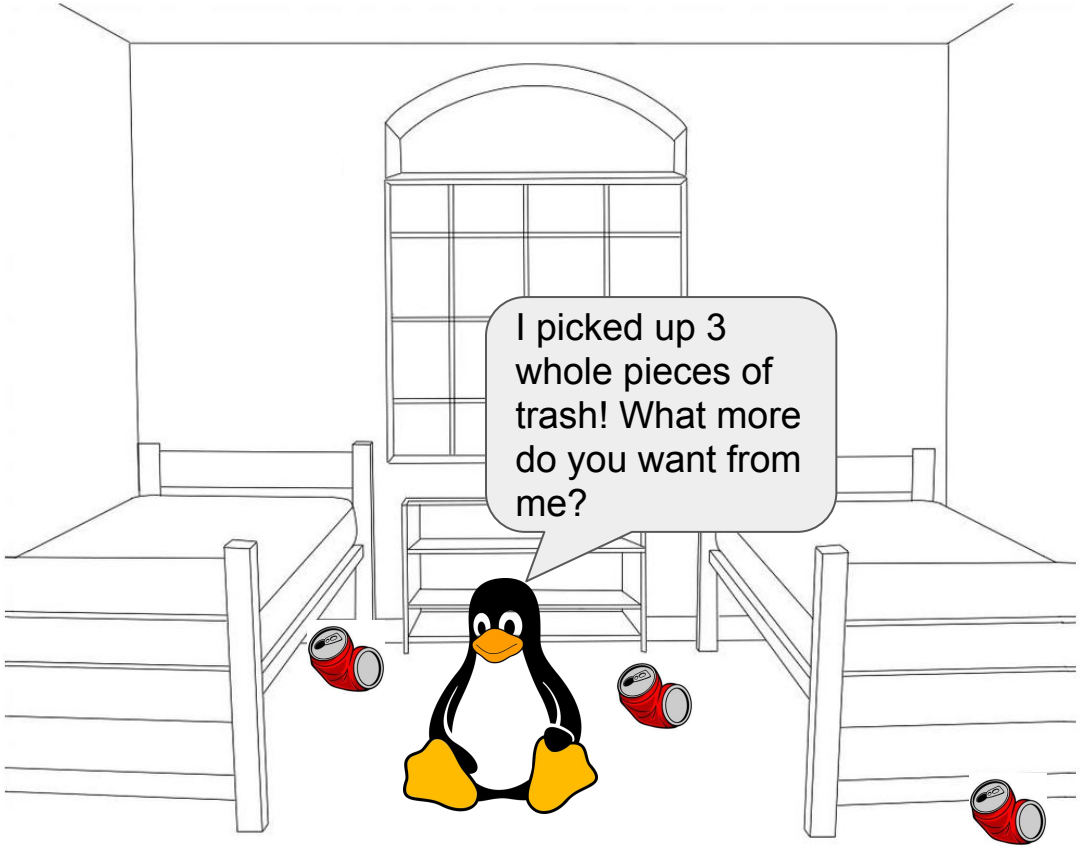


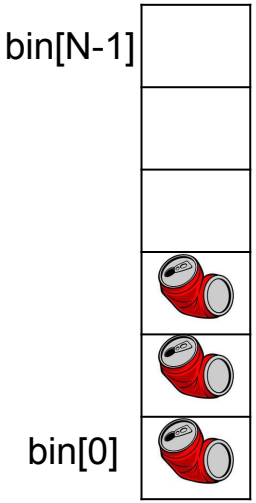
Stop asking them to clean the room! There's nothing to do.

How do we get the room clean?

```
NumTrash pickup(roomNum, trashCan, Amount)
```

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`



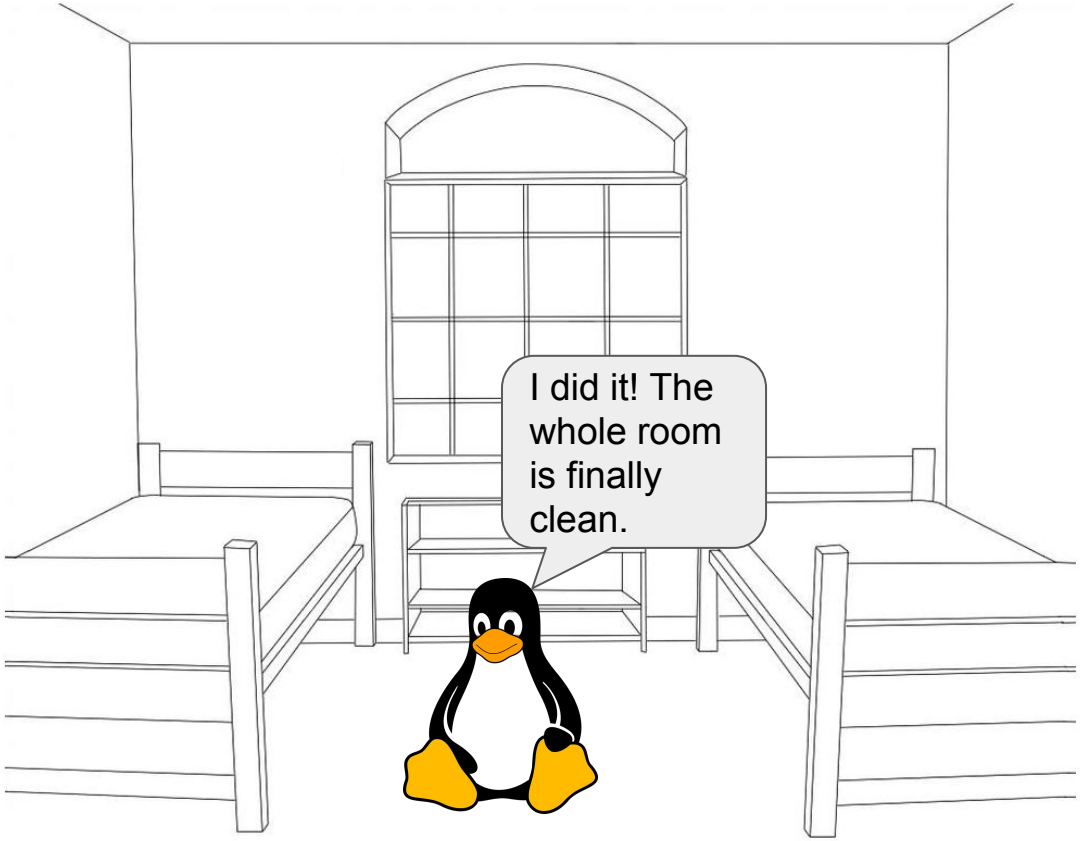


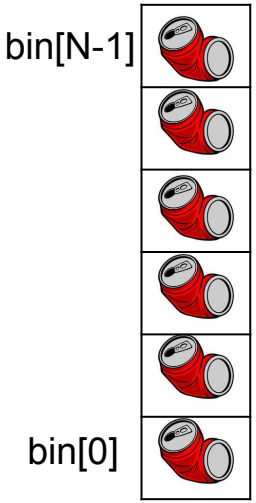
Ask them again to pick up the rest of it.

How do we get the room clean?

```
NumTrash pickup(roomNum, trashCan, Amount)
```

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`





They did what you asked, so stop asking them to pick up trash.

How do we get the room clean?

```
int pickedUp = 0;
while ( ____________ ) {




}
```

`NumTrash pickup(roomNum, trashCan, Amount)`

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`

How do we get the room clean?

NumTrash pickup(roomNum, trashCan, Amount)

NumTrash == -1, errno == excuse
NumTrash == 0
NumTrash < Amount
NumTrash == Amount

```
int pickedUp = 0;
while ( pickedUp < N ) {
    NumTrash = pickup( room, bin + pickedUp, N - pickedUp )
    if ( NumTrash == -1 ) {
        if ( excuse not reasonable )
            ask again
        stop asking and handle the excuse
    }
    if ( NumTrash == 0 )  // we over-estimated the trash
         stop asking since the room is clean
    add NumTrash to pickedUp
}
```

How do we get the room clean?

`NumTrash pickup(roomNum, trashCan, Amount)`

`NumTrash == -1, errno == excuse`
`NumTrash == 0`
`NumTrash < Amount`
`NumTrash == Amount`

```
int pickedUp = 0;
while ( pickedUp < N ) {
    result = pickup( room, bin + pickedUp, N - pickedUp )
    if ( result == -1 ) {
        if ( errno == E_BUSY_NETFLIX )
            continue;
        break;
    }
    if ( result == 0 )
         break;
    pickedUp += result;
}
```

Some Final Notes...

We assumed that there were exactly N pieces of trash (N bytes of data that we wanted to read from a file). How can we modify our solution if we don't know N?

(Answer): Keep trying to `read(...)` until we get 0 back (EOF / clean room)

We determine N dynamically by tracking the number of bytes read until this point, and use `malloc` to allocate more space as we read.

There is no one true loop.
Tailor your POSIX loops to the specifics of what you need!