Section 7 Lab fs

Reminder

- No class on Nov 19th and Nov 24th
- Lab fs due Tuesday Nov 23rd

Special devices

- Appear as files to the user, but do not have corresponding data blocks.
- In this lab, the special devices are also called pseudo devices, in that its read/write is handled not by any physical device but by the kernel.
- What's the use case?
  - /dev/null and /dev/zero are typically passed to programs in place of normal files
  - /dev/null: when we want to discard the output of the program
  - /dev/zero: when we want to create files (or memory) initialized with zeros
  - /dev/random and /dev/uptime are self-explanatory
  - But check out linux /dev/urandom (`man /dev/urandom`) if you're interested: it is a special device, but it gathers environmental noise from physical devices

Creating Special Devices

Recall major and minor device numbers:

- Major number identifies the device driver that handles read/write
- Minor number identifies the exact device

To create a new driver:

- Add new major number to file.h
- On kernel init, modify devsw to map the new major number to the corresponding read/write functions (e.g. see `consoleinit()`).
- Either: 1) create a major number and a read/write function for each special device, or 2) use one single major number and different minor numbers for each special device, and have the single special read/write function examine the provided minor number.

Random number generator

- You may want to use a xorshift generator
- It is a pseudorandom number generator
- Don’t change the parameters. If you change them, the generator may not be able to achieve a long period)

```
uint64_t xorshift64(struct xorshift64_state *state)
{
	uint64_t x = state->a;
	x ^= x << 13;
	x ^= x >> 7;
	x ^= x << 17;
	return state->a = x;
}
```

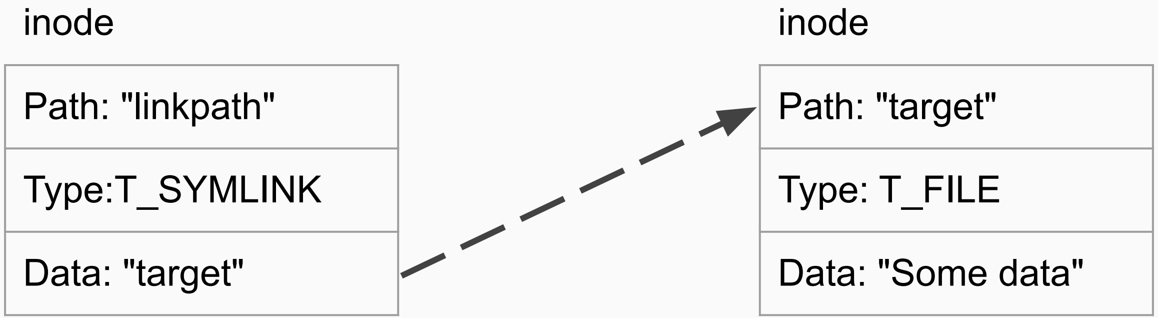
Some other choices

- Fortuna random number generator (more secure)
- LFSR (Linear Feedback Shift Register)
    - Best parameter for LFSR

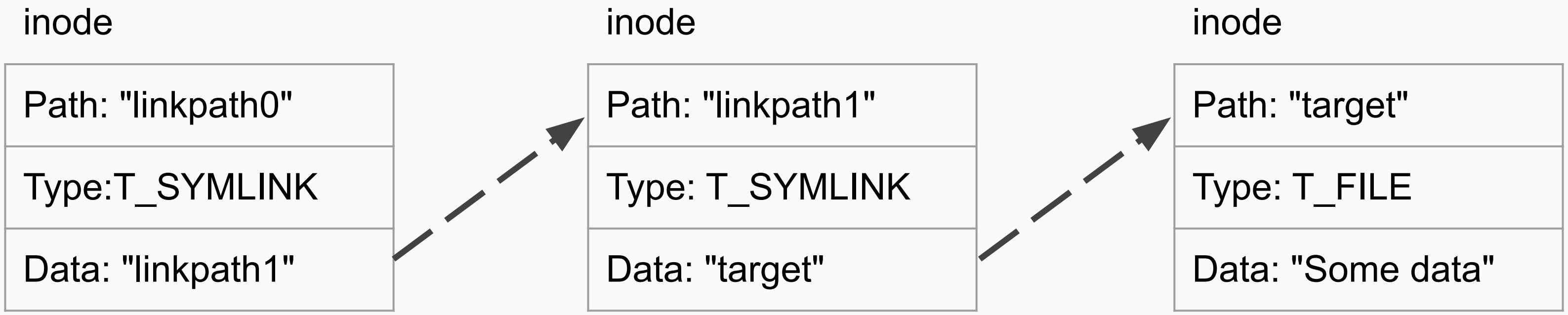
Symbolic links

- A symbolic link is a term for any file that contains a reference to another file

Symbolic links example



Symbolic links recursive example



Symbolic links

- Implement the `symlink(target, linkpath)`
- Make sure `sys_open` support `T_SYMLINK`
- When you are updating the file system, use `begin_op` before any update and `end_op` after update for crash safety (check out e.g. sys_open, sys_mknod for examples). Make sure to add `end_op` before every return.
- Be careful of locks, remember to unlock the inode before return