


RAI 1



Why fsync() on OpenZFS can't fail, and what happens when it does

Rob Norris, Klara Inc.

Hello!

-  Australian
- Klara, Inc.
- OpenZFS developer
- Recovering Linux sysadmin
- FreeBSD non-committer



How to do files

A simple storage service

```
int write_network_stream_to_file(int nfd, char *filename) {
    int err = 0;

    int fd = open(filename, O_CREAT|O_WRONLY|O_TRUNC, S_IRUSR|S_IWUSR);
    if (fd < 0)
        return (errno);

    char buf[1024];
    ssize_t nread;
    while ((nread = read(nfd, buf, sizeof (buf))) > 0) {
        ssize_t nwritten = 0, n;
        while (nwritten < nread && (n = write(fd, &buf[nwritten], sizeof (buf) - nwritten)) >= 0)
            nwritten += n;
        if (n < 0) {
            err = errno;
            close(fd);
            return (err);
        }
    }

    if (fsync(fd) < 0)
        err = errno;

    if (close(fd) < 0 && err == 0)
        err = errno;

    return (err);
}
```



The DMU and the ZIL





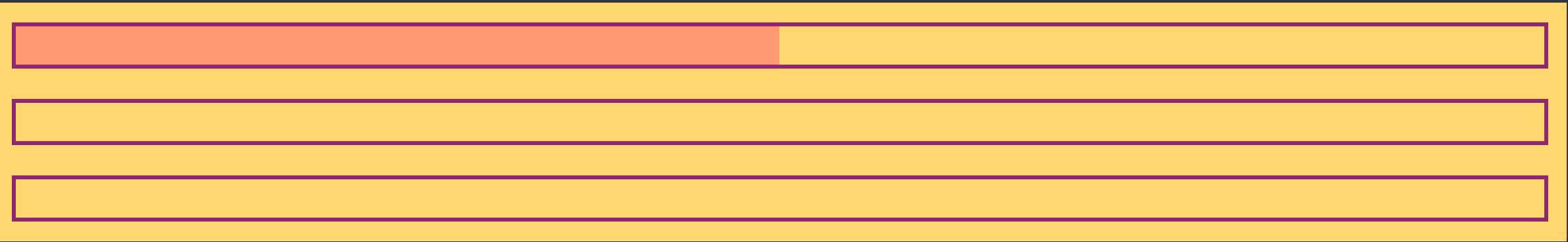
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"..., 1024)
```



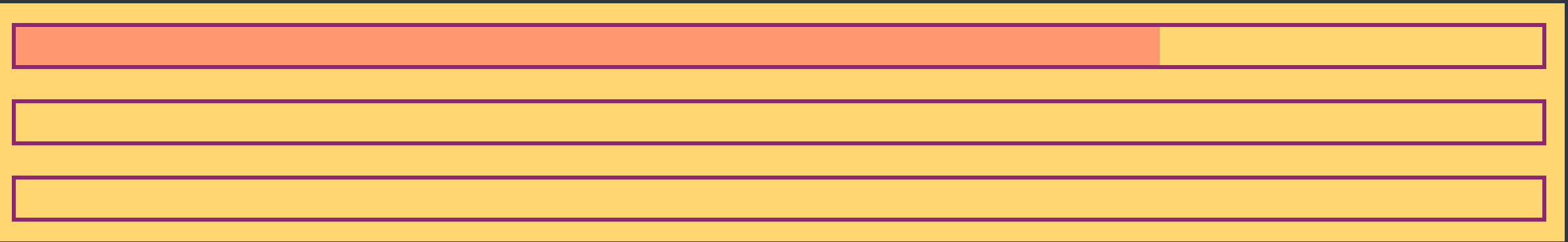
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"..., 1024)  
write(42, "m, corpore alius senescit; Dolor"..., 1024)
```



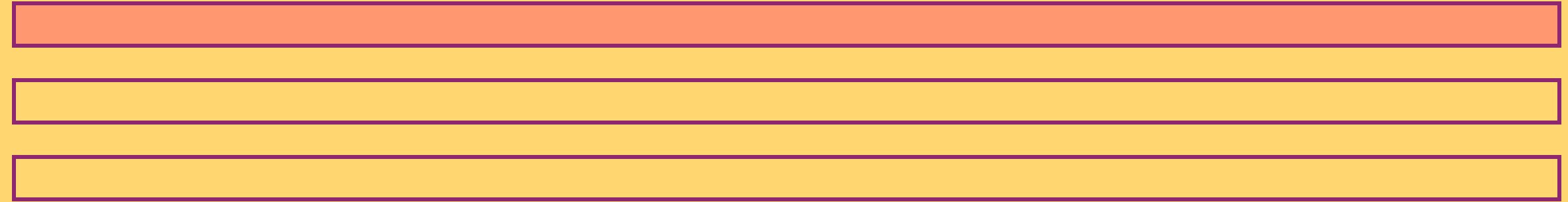

The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)  
write(42, "m, corpore alius senescit; Dolor"... , 1024)  
write(42, " nihil posse ad beatam vitam dee"... , 1024)
```



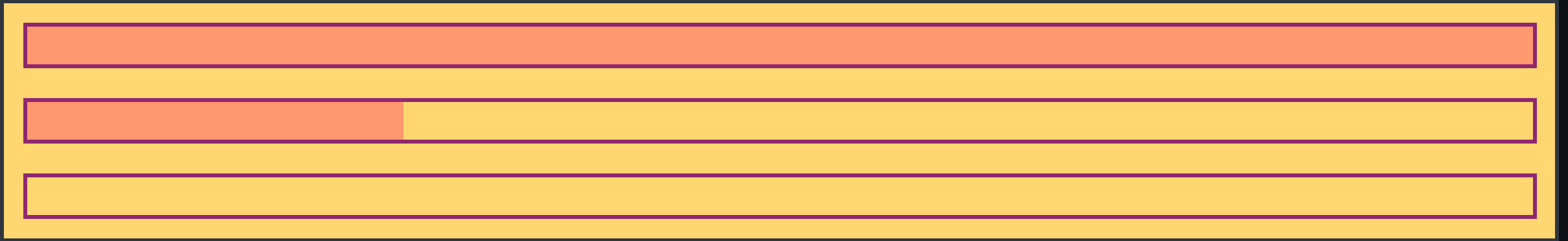
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)  
write(42, "m, corpore alius senescit; Dolor"... , 1024)  
write(42, " nihil posse ad beatam vitam dee"... , 1024)  
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
```



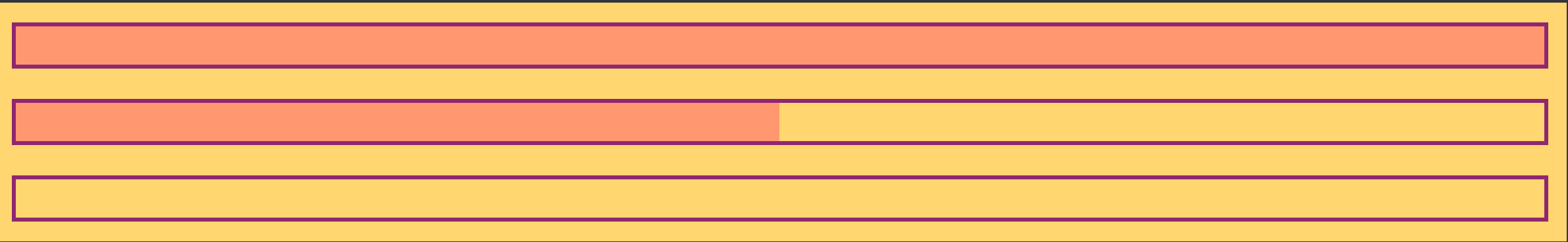
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
```



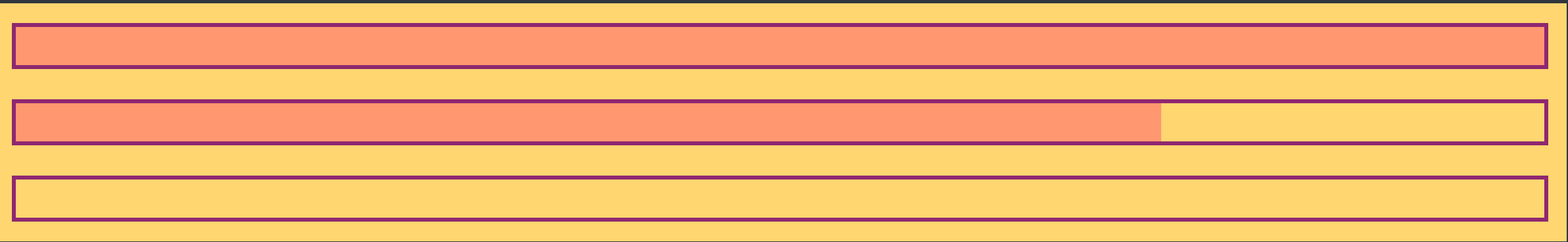
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
```



The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
```

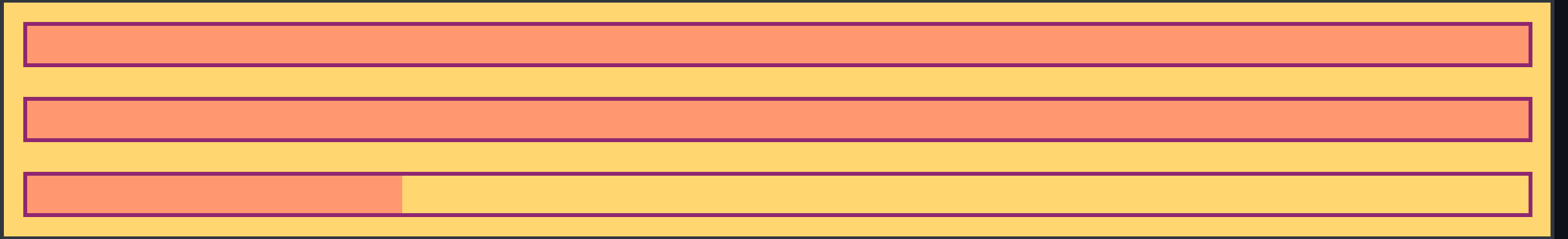


The DMU and the ZIL

```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
```



The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
```

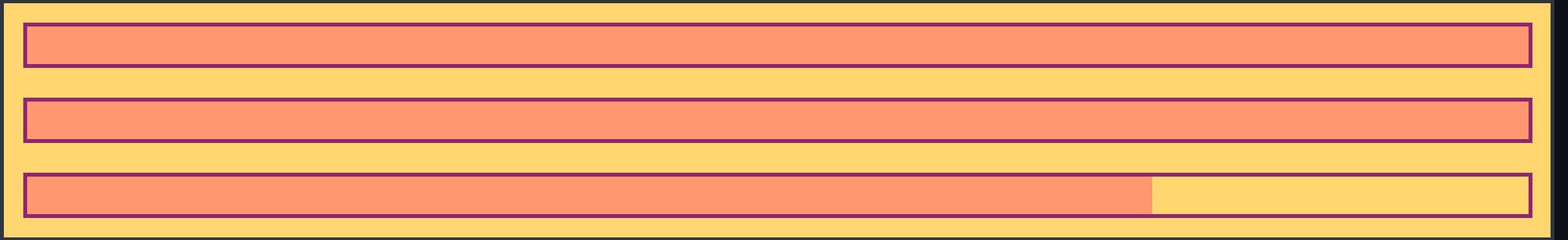


The DMU and the ZIL

```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
```



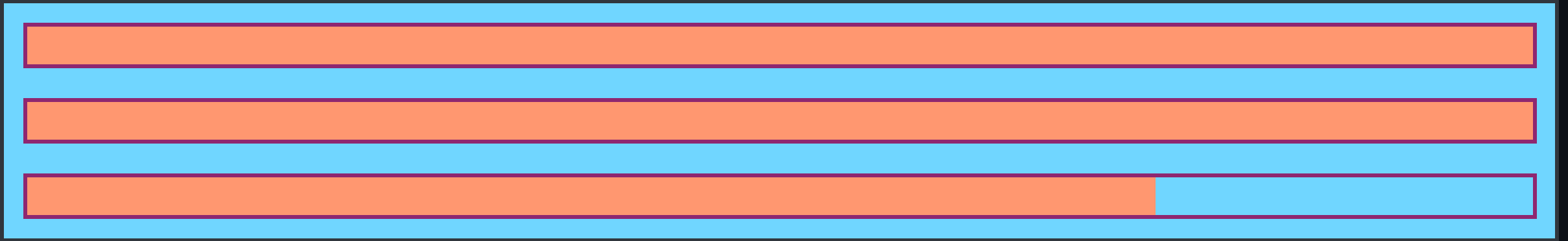

The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
write(42, "istine modo de Carneade? Numquam"... , 1024)
```



The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
write(42, "istine modo de Carneade? Numquam"... , 1024)
```

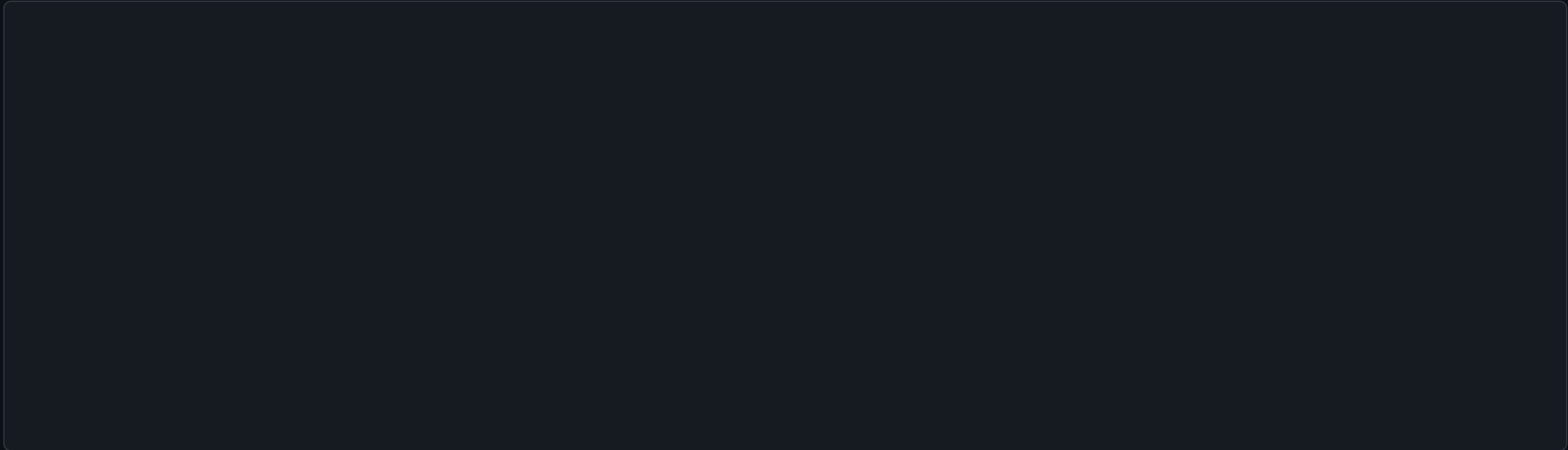
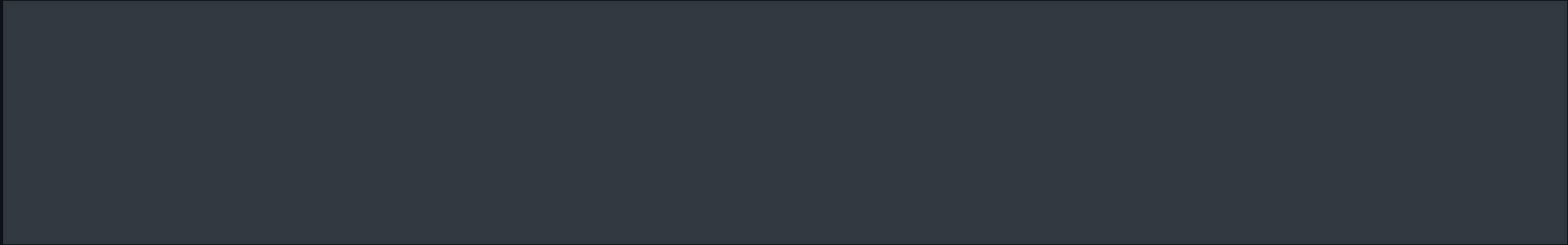


The DMU and the ZIL

```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
write(42, "istine modo de Carneade? Numquam"... , 1024)
```

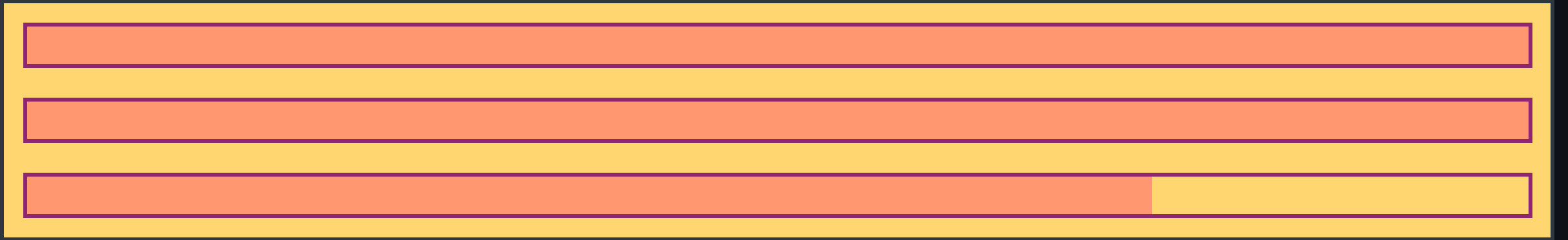


The DMU and the ZIL





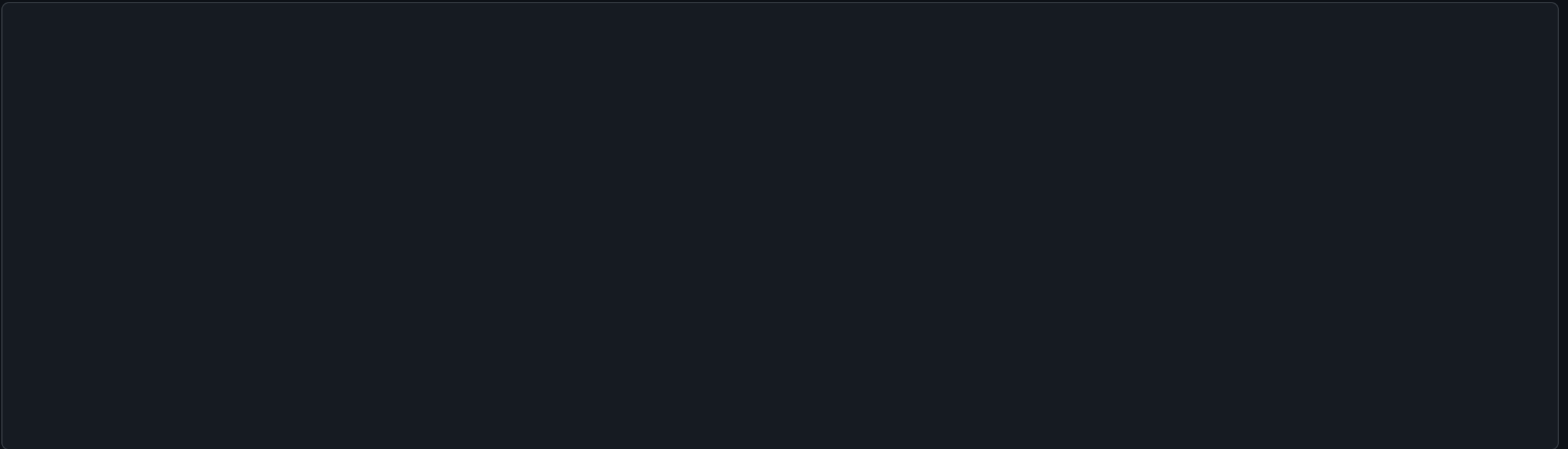
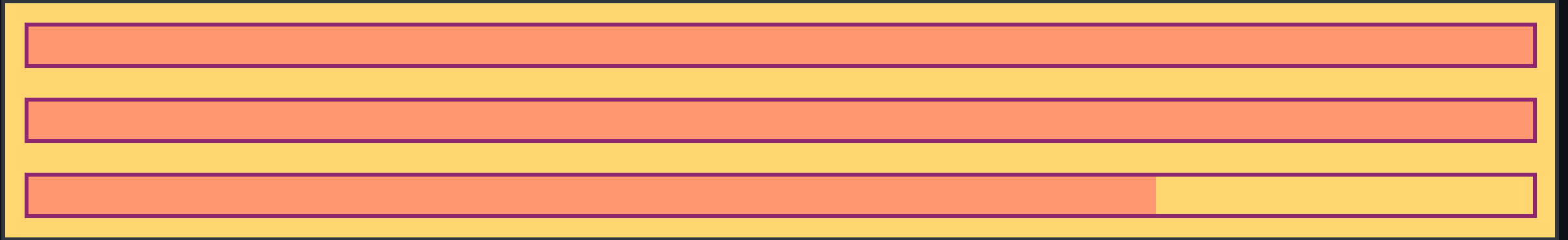
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
write(42, "istine modo de Carneade? Numquam"... , 1024)
```

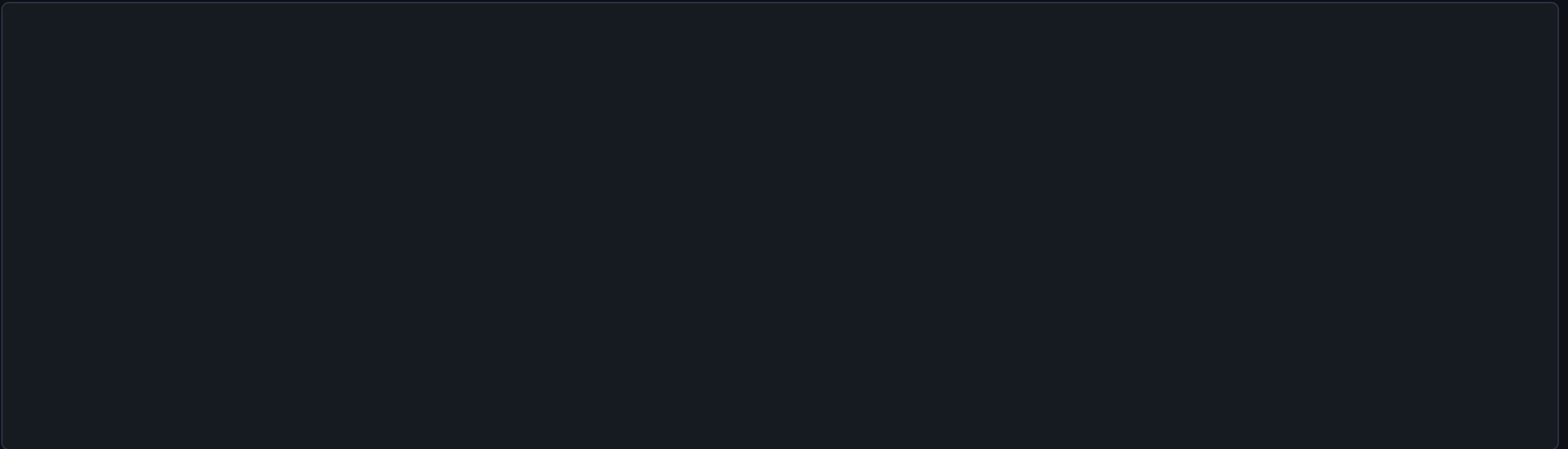
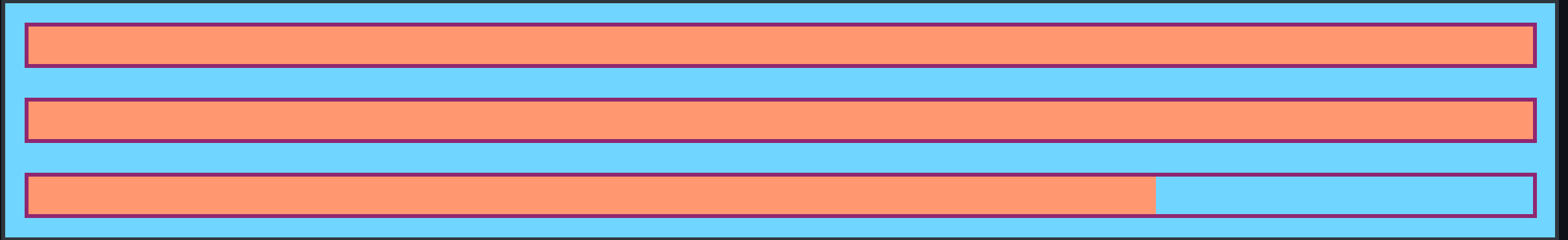


The DMU and the ZIL



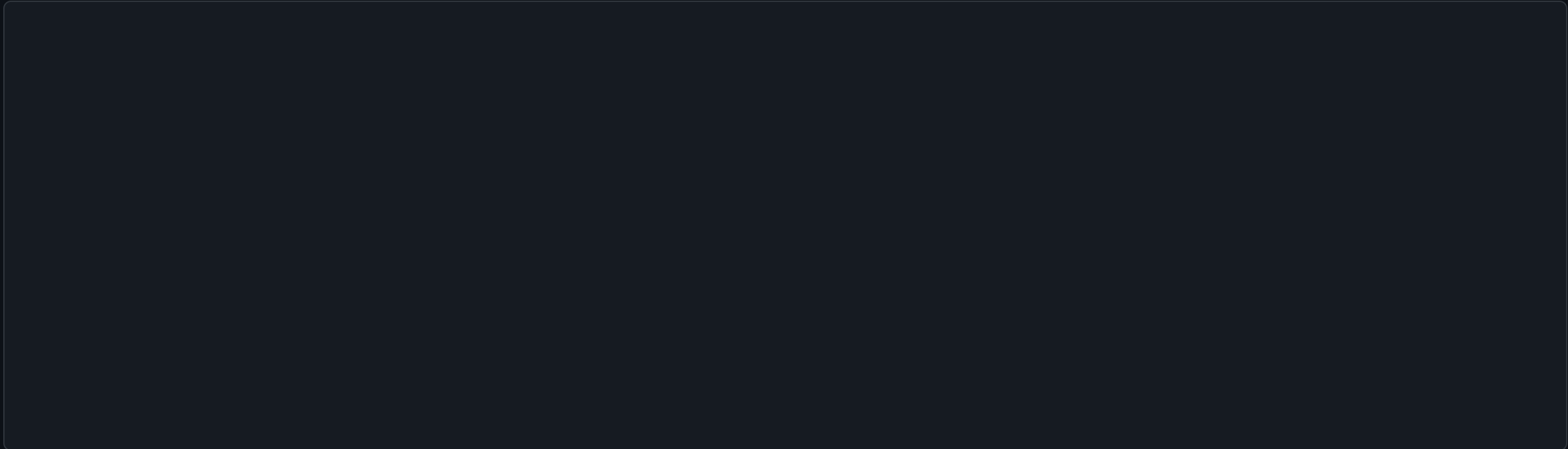
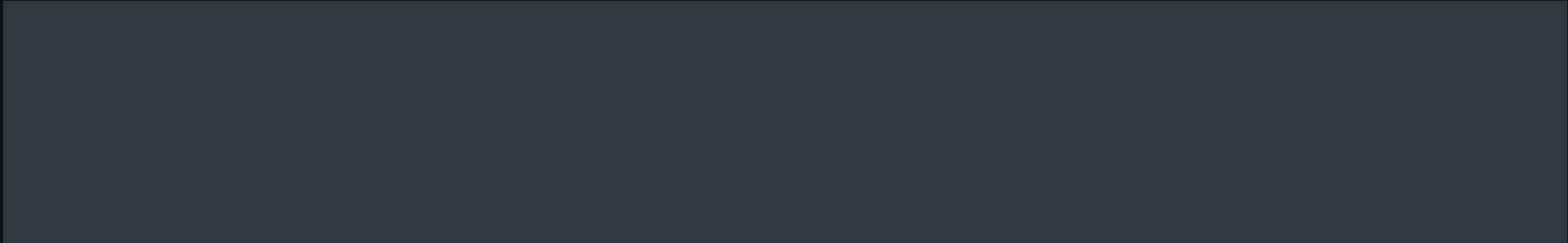


The DMU and the ZIL



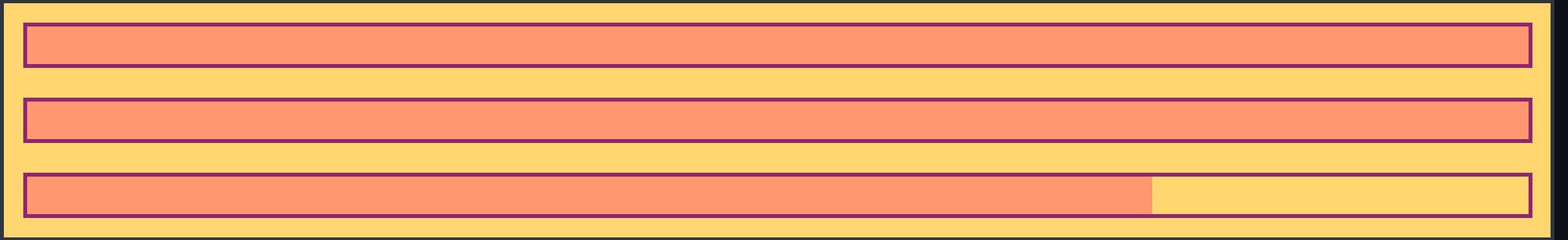


The DMU and the ZIL





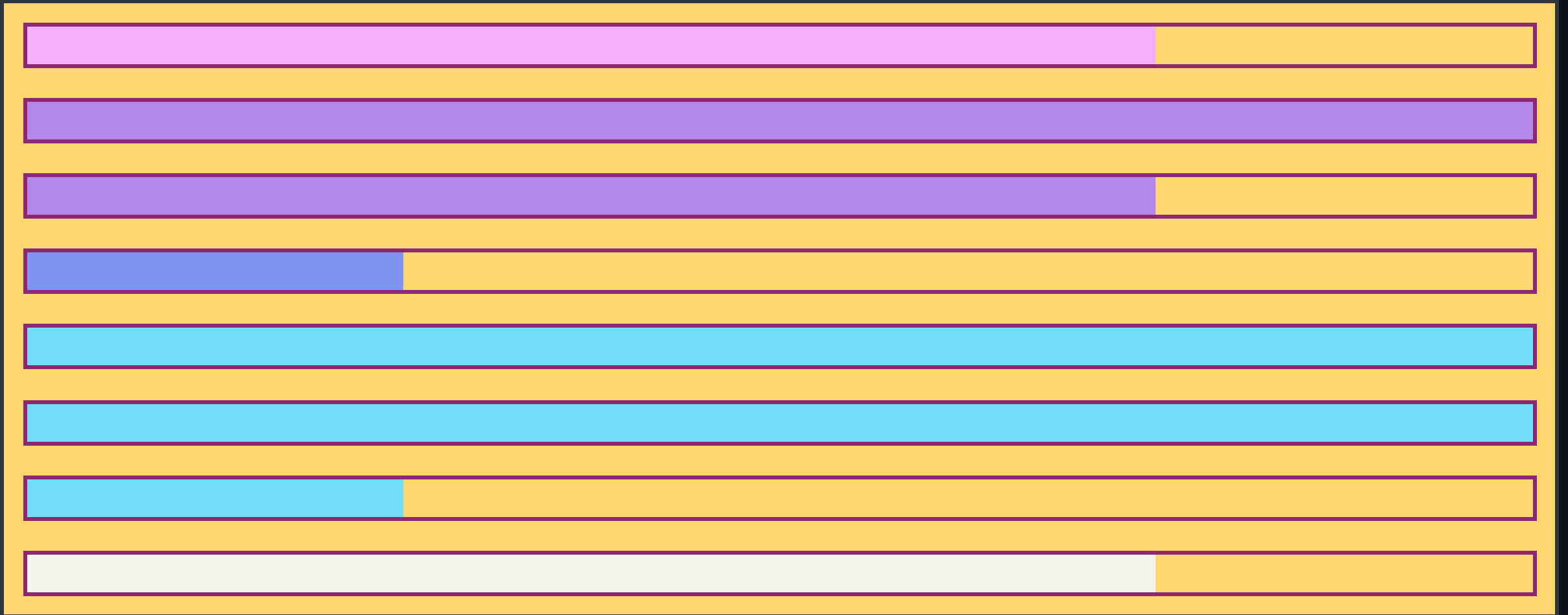
The DMU and the ZIL



```
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(42, "t sit, etiamne post mortem colet"... , 1024)
write(42, "leotes ille Dionysius flagitiose"... , 1024)
write(42, "c haec primum fortasse audientis"... , 1024)
write(42, "istine modo de Carneade? Numquam"... , 1024)
```



The DMU and the ZIL





The DMU and the ZIL

```
write(17, "IT is a truth universally acknow"... , 1024)
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(38, "Ladies and gentleman, well may w"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)
write(7, "ZORK I: The Great Underground Em"... , 1024)
write(42, "unt instituta capienda. In his i"... , 1024)
write(7, "↵You are in the kitchen of the w"... , 1024)
write(16, "When Mr. Bilbo Baggins of Bag En"... , 1024)
write(7, "rn on lamp↵The brass lantern is "... , 1024)
write(7, "lso↵covered with paint). A dark "... , 1024)
write(17, "e from the north of England; tha"... , 1024)
write(17, "ter; for as you are as handsome "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)
write(7, "ding into↵darkness.↵ ↵>go down↵T"... , 1024)
write(42, "scio quem illum anteponebas? Me "... , 1024)
write(7, "his last breath, a cloud of sini"... , 1024)
write(16, "ell as (reputedly)↵inexhaustible"... , 1024)
write(7, "e of the dome (20 feet up) is a↵"... , 1024)
write(7, "ld not get back up it.↵On the tw"... , 1024)
write(7, "le to climb↵down into the canyon"... , 1024)
write(16, "odo was still in his↵tweens, as "... , 1024)
```



The DMU and the ZIL

```
write(17, "IT is a truth universally acknow"... , 1024)
write(42, "Lorem ipsum dolor sit amet, cons"... , 1024)
write(42, "m, corpore alius senescit; Dolor"... , 1024)
write(42, " nihil posse ad beatam vitam dee"... , 1024)
write(38, "Ladies and gentleman, well may w"... , 1024)
write(42, "m cum medicinam pollicetur, luxu"... , 1024)

write(42, "unt instituta capienda. In his i"... , 1024)

write(16, "When Mr. Bilbo Baggins of Bag En"... , 1024)

write(17, "e from the north of England; tha"... , 1024)
write(17, "ter; for as you are as handsome "... , 1024)
write(42, "st, nunc quidem hactenus; Quae q"... , 1024)

write(42, "scio quem illum anteponebas? Me "... , 1024)

write(16, "ell as (reputedly)inexhaustible"... , 1024)

write(16, "odo was still in histweens, as "... , 1024)
```

🤔 Writing files: a review

- Changes recorded in two ways
 - DMU: **what** we changed (latest state)
 - ZIL: **how** we changed it
- DMU buffers are bound to a transaction
 - Do nothing, the transaction closes, and is written out atomically
- ZIL contents can be written out by object, on demand
 - `fsync()`

`fsync()`





`fsync()`



FSYNC





`fsync()`

never fails



`fsync()` never fails

```
int
zfs_fsync(znode_t *zp)
{
    zfsvfs_t *zfsvfs = ZTOZSB(zp);

    zil_commit(zfsvfs->z_log, zp->z_id);

    return (0);
}
```



`fsync()` never fails

```
int
zfs_fsync(znode_t *zp)
{
    zfsvfs_t *zfsvfs = ZTOZSB(zp);

    zil_commit(zfsvfs->z_log, zp->z_id);

    return (0);
}

void
zil_commit(zilog_t *zilog, uint64_t foid)
{
    ...
}
```



`fsync()` never fails

```
int
zfs_fsync(znode_t *zp)
{
    zfsvfs_t *zfsvfs = ZTOZSB(zp);

    zil_commit(zfsvfs->z_log, zp->z_id);

    return (0);
}

void
zil_commit(zilog_t *zilog, uint64_t foid)
{
    int err = zil_commit_wait(zilog, foid);
    if (err != 0)
        txg_wait_synced(zilog->zl_dmu_pool, 0);
}
```



`fsync()`

never fails

✓ `fsync()` never fails
(but can take a very long lunch 🍔 🍟)

All stop

- Pool has suspended
- `fsync()` is blocked
- Application is waiting
- User connection is waiting

All stop

- `fsync()` *could* return an error
- Application could redirect request to another machine/pool/shard/etc
- User service continues



Failing with style

- `failmode` pool property
 - `wait`: block until the pool returns (default)
 - `panic`: panic the kernel 💀
 - `continue`:
 - new write ops: `EIO`
 - in-flight sync ops: block
 - read ops: ??? (cached, surviving disks can service, etc)



Our mission



Our mission

(should we choose to accept it)



Our mission

(should we choose to accept it)

(which we will, because a customer is paying for it)

Pay day

If the pool suspends, any `fsync()` in progress should return `EIO`.

Sync ops

- `fsync(fd)`
- `fdatasync(fd)`
- `sync()`
- `syncfs(fd)` (Linux)
- `msync(MS_SYNC)`
- `write()` / `writev()` / `pwritev()` after `open(O_SYNC|O_DSYNC)`
- `pwritev2(RWF_SYNC/RWF_DSYNC)` (Linux)
- `sync_file_range()` (Linux)
- aio (`aio_fsync`, `aio_write`, etc)
- io_uring (`IORING_OP_FSYNC`, `IORING_OP_WRITEV`, etc) (Linux)
- `sync=always`



Pay day

If the pool suspends, any call blocked in `zil_commit()` should return an appropriate error.



How to draw an owl

```
diff --git include/sys/zil.h include/sys/zil.h
index 4747ecc06..3b7bb8ed4 100644
--- include/sys/zil.h
+++ include/sys/zil.h
@@ -571,7 +571,7 @@ extern void zil_itx_destroy(itx_t *itx);
extern void      zil_itx_assign(zilog_t *zilog, itx_t *itx, dmu_tx_t *tx);

extern void      zil_async_to_sync(zilog_t *zilog, uint64_t oid);
-extern void      zil_commit(zilog_t *zilog, uint64_t oid);
+extern int       zil_commit(zilog_t *zilog, uint64_t oid);
extern void      zil_commit_impl(zilog_t *zilog, uint64_t oid);
extern void      zil_remove_async(zilog_t *zilog, uint64_t oid);

diff --git module/zfs/zil.c module/zfs/zil.c
index 34be54b33..52b18b8e4 100644
--- module/zfs/zil.c
+++ module/zfs/zil.c
@@ -3548,7 +3548,7 @@ zil_commit_itx_assign(zilog_t *zilog, zil_commit_waiter_t *zicw)
 *      but the order in which they complete will be the same order in
 *      which they were created.
 */
-void
+int
zil_commit(zilog_t *zilog, uint64_t foid)
{
    /*
```





How to draw an owl

```
diff --git module/zfs/zfs_vnops.c module/zfs/zfs_vnops.c
index babb07ca2..17ac19bb5 100644
--- module/zfs/zfs_vnops.c
+++ module/zfs/zfs_vnops.c
@@ -89,7 +89,7 @@ zfs_fsync(znode_t *zp, int syncflag, cred_t *cr)
     if ((error = zfs_enter_verify_zp(zfsvfs, zp, FTAG)) != 0)
         return (error);
     atomic_inc_32(&zp->z_sync_writes_cnt);
-    zil_commit(zfsvfs->z_log, zp->z_id);
+    error = zil_commit(zfsvfs->z_log, zp->z_id);
     atomic_dec_32(&zp->z_sync_writes_cnt);
     zfs_exit(zfsvfs, FTAG);
 }
```





Draw the rest of
the f.....g owl

⚠ ZIL fallback sync

```
void
zil_commit(zilog_t *zilog, uint64_t foid)
{
    int err = zil_commit_wait(zilog, foid);
    if (err != 0)
        txg_wait_synced(zilog->zl_dmu_pool, 0);
}
```

Transaction sync

```
void
txg_wait_synced(dsl_pool_t *dp, uint64_t txg)
{
    VERIFY0(txg_wait_synced_impl(dp, txg, B_FALSE));
}

boolean_t
txg_wait_synced_sig(dsl_pool_t *dp, uint64_t txg)
{
    return (txg_wait_synced_impl(dp, txg, B_TRUE));
}
```

Transaction sync

```
static boolean_t
txg_wait_synced_impl(dsl_pool_t *dp, uint64_t txg, boolean_t wait_sig)
{
    tx_state_t *tx = &dp->dp_tx;
    mutex_enter(&tx->tx_sync_lock);
    while (tx->tx_synced_txg < txg) {
        cv_broadcast(&tx->tx_sync_more_cv);
        if (wait_sig) {
            if (cv_wait_io_sig(&tx->tx_sync_done_cv,
                &tx->tx_sync_lock) == 0) {
                mutex_exit(&tx->tx_sync_lock);
                return (B_TRUE);
            }
        } else {
            cv_wait_io(&tx->tx_sync_done_cv, &tx->tx_sync_lock);
        }
    }
    mutex_exit(&tx->tx_sync_lock);
    return (B_FALSE);
}
```



It's

complicated

Support forced export of ZFS pools #11082

 Open

wca wants to merge 1 commit into `openzfs:master` from `KlaraSystems:forced-export` 

 Conversation 141

 Commits 1

 Checks 19

 Files changed 91



wca commented on Oct 17, 2020 • edited ▾

Contributor ...

Motivation and Context

This change enables users to forcibly export a pool in a manner which safely discards all pending dirty data, transaction groups, associated zios, metaslab changes, etc. It allows a regular `zpool export` to fail out so that it releases the namespace lock to enable a forced export. It is able to do this regardless of whether the current pool activity involves send, receive, or POSIX I/O.

This allows users to continue using their system without rebooting, in the event the disk cannot be recovered in an online manner, or the user prefers to resume use of their pool at a later time. Since ZFS can simply resume from the most recent consistent transaction group, the latter is easily achieved.

Closes [#3461](#)

Forced export

```
typedef enum {
    /* No special wait flags. */
    TXG_WAIT_F_NONE          = 0,
    /* Reject the call with EINTR upon receiving a signal. */
    TXG_WAIT_F_SIGNAL       = (1U << 0),
    /* Reject the call with EAGAIN upon suspension. */
    TXG_WAIT_F_NOSUSPEND   = (1U << 1),
    /* Ignore errors and export anyway. */
    TXG_WAIT_F_FORCE_EXPORT = (1U << 2),
} txg_wait_flag_t;

int
txg_wait_synced_flags(dsl_pool_t *dp, uint64_t txg, txg_wait_flag_t flags)
{
    ...
}
```


⚠ ZIL fallback sync

```
void
zil_commit(zilog_t *zilog, uint64_t foid)
{
    int err = zil_commit_wait(zilog, foid);
    if (err != 0)
        txg_wait_synced(zilog->zl_dmu_pool, 0);
}
```

⚠ ZIL fallback sync

```
int
zil_commit(zilog_t *zilog, uint64_t foid)
{
    int err = zil_commit_wait(zilog, foid);
    if (err != 0) {
        err = txg_wait_synced_flags(zilog->zl_dmu_pool, 0,
            TXG_WAIT_F_NOSUSPEND);
        if (err == EAGAIN)
            err = SET_ERROR(EIO);
    }
    return (err);
}
```

I BELIEVED THE
FALSEHOODS YOU
TOLD ME

BECAUSE
TRUST!



NATHANWPYLE



Forbidden knowledge



★ fsyncgate

- postgres does async writes, which go to the page cache
 - kernel page flush begins. Some writes fail
 - kernel sets an *failed* flag on those pages, but can't inform application
- postgres begins writing a checkpoint, calls `fsync()`
 - kernel returns `EIO` because there are *failed* pages
 - kernel clears the *failed* and *dirty* flags
- postgres aborts the checkpoint
- postgres does async writes, begins a new checkpoint, calls `fsync()`
- more-recently-dirtied pages are flushed, `fsync()` returns success

Royal decree

POSIX (IEEE Std 1003.1-2017):

The `fsync()` function shall request that all data for the open file descriptor named by `filides` is to be transferred to the storage device associated with the file described by `filides`. The nature of the transfer is implementation-defined. The `fsync()` function shall not return until the system has completed that action or until an error is detected.

Royal decree

POSIX (IEEE Std 1003.1-2017):

The `fsync()` function shall request that all data for the open file descriptor named by `fdes` is to be transferred to the storage device associated with the file described by `fdes`. The nature of the transfer is implementation-defined. The `fsync()` function shall not return until the system has completed that action or until an error is detected.

If the `fsync()` function fails, outstanding I/O operations are not guaranteed to have been completed.

`EIO`: An I/O error occurred while reading from or writing to the file system.

★ Page flags

- Three page flags:
 - *dirty*: needs to be written out
 - *error*: last write attempt failed
 - *invalid*: page is unuseable and can be freed

★ Page flags: FreeBSD < 4

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error: sets *invalid* (page unuseable, will be freed)

★ Page flags: FreeBSD ≥ 4

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error: clears *error*, leaves *dirty* set (will retry)

★ Page flags: FreeBSD ≥ 12

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error:
 - `ENXIO`: sets *invalid* (page unuseable, will be freed)
 - other errors: clears *error*, leaves *dirty* set (will retry)

★ Page flags: Linux

- flush failure: sets page *error* flag, clears *dirty* flag
- `fsync()` error: clears *error* flag (**NOT INVALIDATED**)
 - this is weird

★ Page flags: DragonflyBSD

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error: clears *error*, leaves *dirty* set (will retry)

★ Page flags: NetBSD

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error: sets *invalid* (page unuseable, will be freed)

★ Page flags: OpenBSD

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error:
 - sets *invalid* (page unuseable, will be freed)
 - marks the vnode *damaged*
 - all future `fsync()` calls return `EIO` until the vnode is freed (that is, all file descriptors closed)

★ Page flags: macOS (Darwin/XNU)

- flush failure: sets page *error* flag, leaves *dirty* flag set
- `fsync()` error: sets *invalid* (page unuseable, will be freed)
- (but maybe other stuff because 🍏 ✨ ©™®)

★ Page flags: Illumos

- flush failure: sets page *error* flag, clears *dirty* flag
- `fsync()` error: clears *error* flag, leaves *dirty* set (will retry)
 - if the flush fails again, puts page on free list with *delayed write* flag set
 - (I confess I do not fully understand this)



Application response

★ Application response: explicit abort/panic

- PostgreSQL
- MySQL (InnoDB)
- MongoDB (WiredTiger)

★ Application response: assume success

- SQLite
- Xapian

★ Application response: attempt to recover

- Redis (with AOF)
- Cyrus

💥 Application response: generic IO error



EIO

- no uniformity, no expectations
 - we can do what we want!
- question: keep data dirty and retry, or invalidate and free?
- or: is this failure transient or permanent?
 - transient: pool will unsuspend soon
 - permanent: pool will *never* return
- decision: treat errors as transient
 - ZFS pools are usually planned and managed
 - it probably *is* coming back; we should wait.

Status report

- In production at a customer site
- Needs to be forward-ported from 2.1.5
- Forced export needs to be finished and merged
- Add a new `failmode=error` to enable this behaviour
- Needs heavy testing; very invasive in the ZIL code
- Hope to finish and land before end of 2024

Future work

- `failmode=readonly`
 - all writes return `EROFS`
 - may help applications avoid their recovery codepaths unnecessarily

The logo features a stylized 'F' on the left, composed of a red star with three horizontal lines extending to the left, and a light blue 'F' shape. To the right of this is the word 'SYNC' in a bold, white, sans-serif font with a black outline. The entire logo is set against a dark background with a glowing orange and yellow aura.

F5SYNC