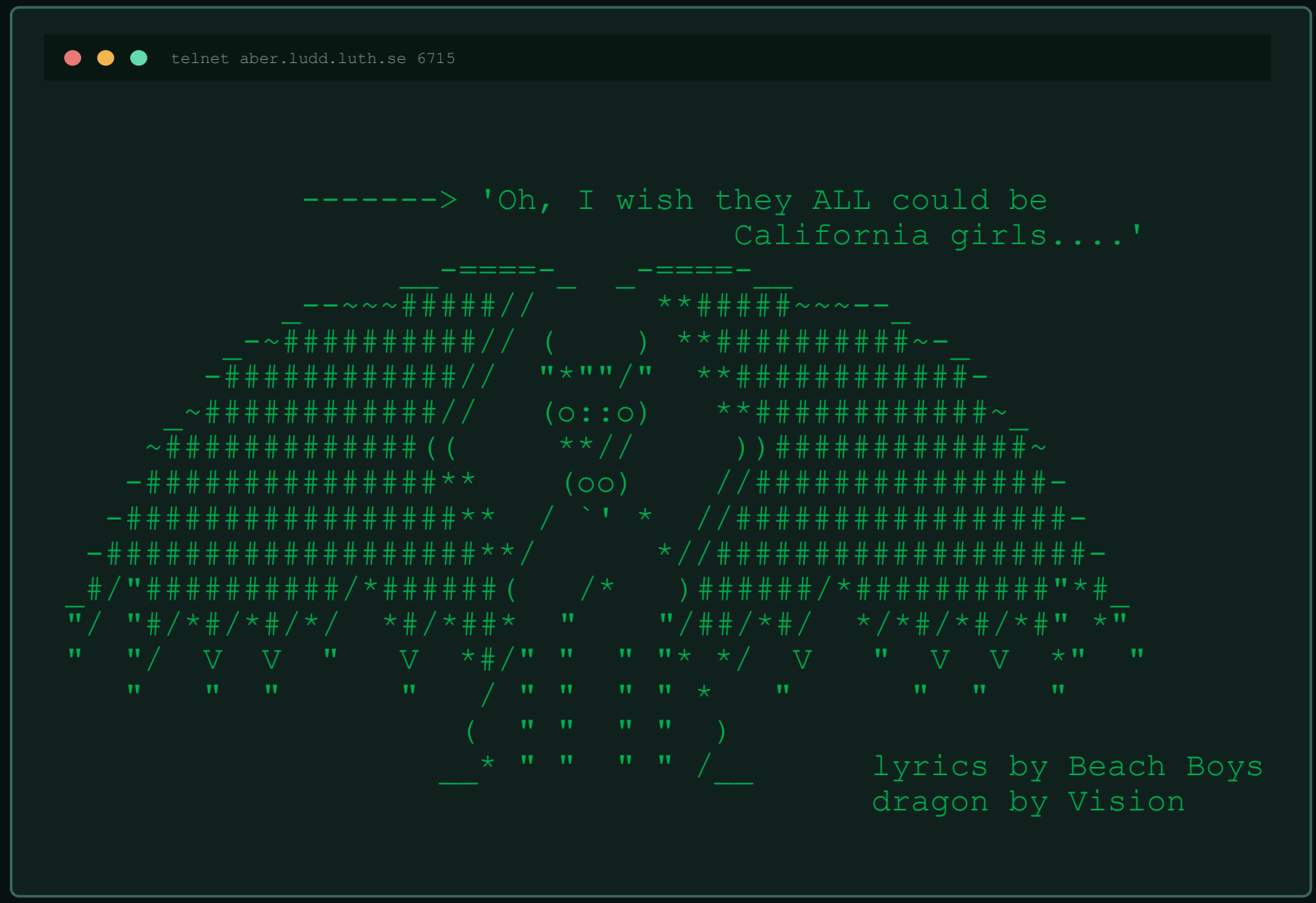
FROM DIRT TO PYTHON

Restoring a 1990s shared world, and looking towards ‘porting’ to modern Python.

telnet mud.cademuir.eu 6715

Albert White • **PyCon Dublin**



albertw@sionnach: ~/projects, | Ubuntu

```
The Village Church
   You are in the village's small wooden church.  A gentle breeze blows into
the church disturbing the dust which dances in the sunbeams that shine faintly
through the windows.  A doorway leads south.
   At your feet a huge sacrificial pit allows you to give valuables to the gods
in the hope of being rewarded.
Seamas The Leprechaun eyes you cautiously, an evil grin upon his face.

>examine seamas
A typical, run of the mill Seamas.
>exits
Obvious exits are:
South : The Village Green
>s
The Village Green
   You are standing on a square of grass in the center of the village.  A road
leads east into the rest of the village.
   To the north is a small church, and a path leads west into the trees.

>w
The Humpbacked Bridge
   You are standing on a crumbling humpbacked bridge looking over the side down
onto a wide but shallow stream.  The sun sparkles gently off the ripples as the
stream passes across pebbles and rocks and the fish swim between them. You hear
the faint roar of a waterfall somewhere downstream.
   You can go east or west across the bridge or follow the stream southwards.
A sign has been posted by the bridge.

>
```

What is a Multi User Dungeon?

AS A GAME

A persistent text adventure made from rooms, exits, objects, creatures, commands and quests. Players type short instructions such as LOOK, NORTH, GET SWORD or SAY HELLO.

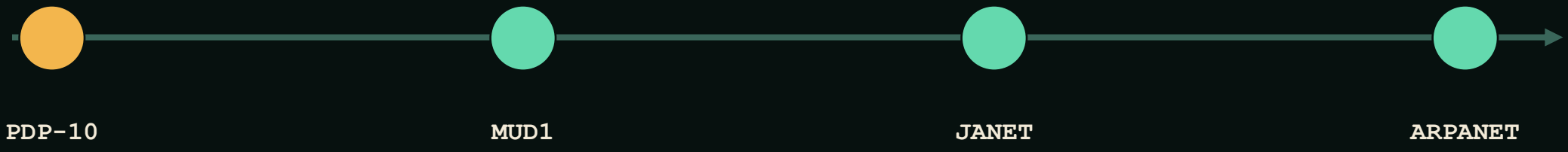
AS A SYSTEM

A network server that accepts many connections, tracks shared state, parses commands and periodically advances the world. It is part game engine, part database and part online community.

The first age: MUD1

Roy Trubshaw and Richard Bartle wrote Multi-User Dungeon 1 on a DEC PDP-10. The name “dungeon” came from DUNGEN, a FORTRAN port of Zork that was available at Essex, rather than directly from Dungeons & Dragons.

Copies of MUDs began appearing in Universities, though many did not allow external access.



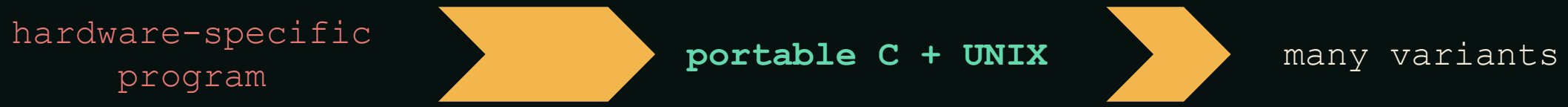
Richard Bartle. **Designing Virtual Worlds**. mud.co.uk/richard/DesigningVirtualWorlds.pdf

The second age: portability changes everything

Many MUDs were individual projects. AberMUD was different: Alan Cox began it at Aberystwyth University, first in B and then in C. The C version could run under UNIX instead of depending on one particular mainframe.

WHY C MATTERED

Source code could move between universities and UNIX systems. That portability encouraged forks, variants and local worlds. It was not tied to particular hardware.



The third age: DIRT and Northern Lights

AberMUD 4 was extended into DIRT MUD 1 around 1990 by Alf Salte (Alf) and G Sorseth (Nicknack). It then evolved into Northern Lights in 1992, operated by the computer society at Luleå University of Technology, Sweden. Northern Lights became one of the better-known MUDs of the 1990s.

UCD Netsoc also had a MUD in the late 1990's – Eiru. It was based on the CircleMUD codebase.



By this point, MUDs were equal parts game and social space: bulletin boards, quests, friendships, player-created areas, wizard roles and even in-world weddings.

Northern Lights

MUD’s were active places through the 1990’s but popularity waned.

Northern Lights remained online for more than 30 years, largely through the work of Anna ‘Vitastjern’ Eklund. By its final years it mainly served nostalgia, but the world was still there and still reachable.

AROUND 2024

A catastrophic failure of the old server meant it would not return.

“NL existed for 31+ years. Let it rest now.”

Some zone authors had asked that their work not be released, and those wishes mattered more than the technical possibility of reconstruction.

While Northern Lights itself was not available for revival, the authors of the older DIRT 3.1 source code (Alf and Nicknack) agreed to send the code and have it hosted on github.

The restoration project

The initial goal was deliberately narrow: compile the original C source on a modern Linux system, start the server, load the original world files and allow several users to log in and play.
Follow archaeological guidelines – minimal intervention.



That sounds modest, but the code carries assumptions from early-1990s UNIX and C environments.

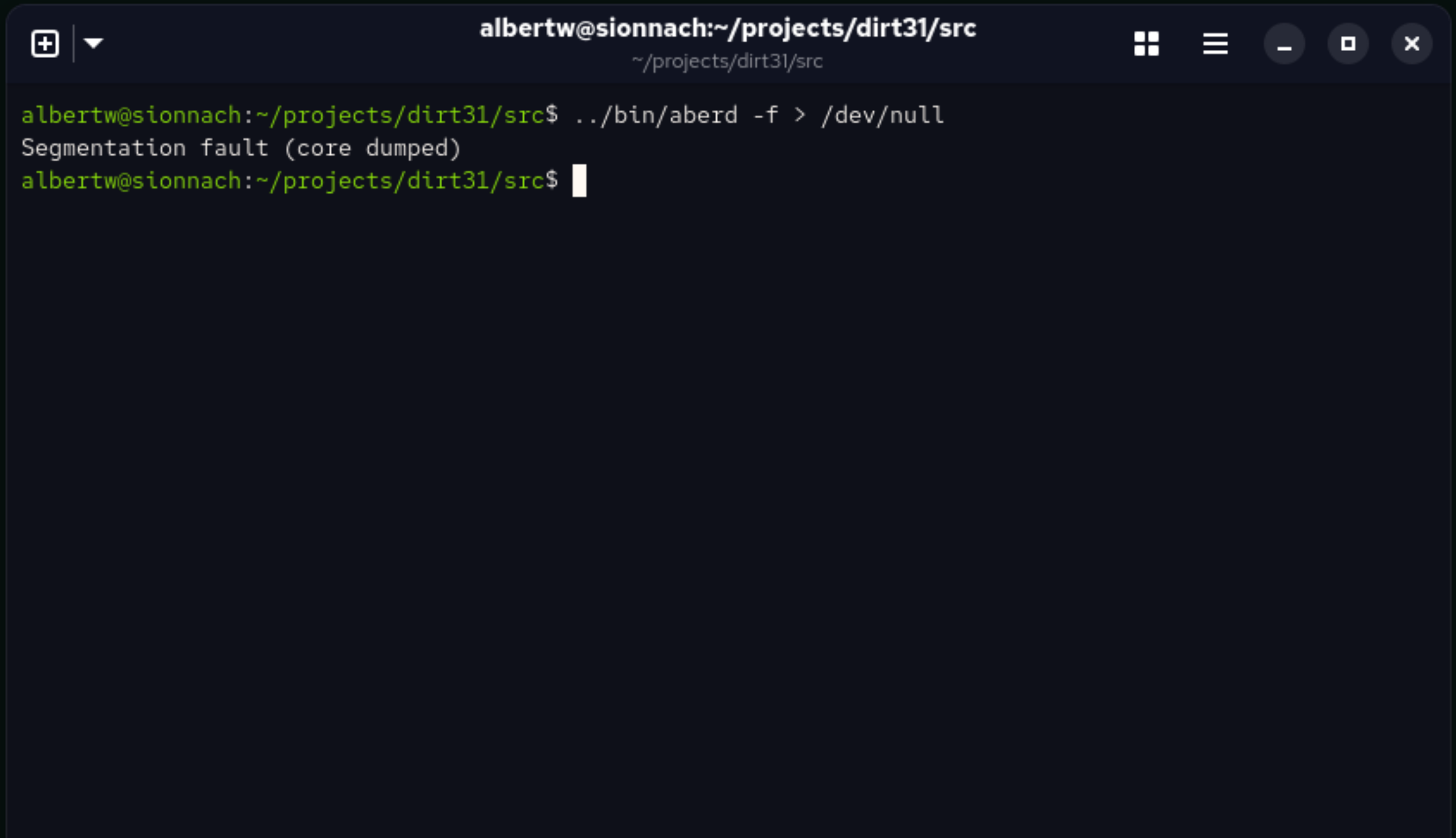
Old C remembers old systems

legacy.c

```
1 extern char *sys_errlist[];
2 extern int errno;
3
4 error(sys_errlist[errno]);
5 exit(1);
```

The source predates assumptions that now feel normal. ANSI C (1989) introduced stdlib.h which declared exit(), and standardized errno via error.h. Over time the old ways stopped being supported.

- Include modern standard headers
- Use errno.h and strerror()
- Change only what prevents building or running


albertw@sionnach:~/projects/dirt31/src
~/projects/dirt31/src
albertw@sionnach:~/projects/dirt31/src$ ../bin/aberd -f > /dev/null
Segmentation fault (core dumped)
albertw@sionnach:~/projects/dirt31/src$

albertw@sionnach:~/projects/dirt31/src — gdb -q --args ../bin/aberd -f

~/projects/dirt31/src

```
A total of 646860 bytes used.
Connected to port 6715 on sionnach.

Program received signal SIGSEGV, Segmentation fault.
check_send_msg (plx=plx@entry=111, a=a@entry=-13792,
    t=t@entry=0x7fffffffd240 "The Thief has gone west.\n") at sendsys.c:172
172         if (test_rcv(plx,b->mode,b->min,b->max,b->x1,b->x2)) return t;
(gdb) bt
#0  check_send_msg (plx=plx@entry=111, a=a@entry=-13792,
    t=t@entry=0x7fffffffd240 "The Thief has gone west.\n") at sendsys.c:172
#1  0x0000000000427418 in send_g_msg (destination=destination@entry=-613,
    func=func@entry=0x426f6b <check_send_msg>, arg=arg@entry=-13792,
    text=text@entry=0x7fffffffd240 "The Thief has gone west.\n") at sendsys.c:151
#2  0x0000000000427526 in send_msg (destination=-613, mode=mode@entry=0, min=<optimized out>,
    max=max@entry=100000, x1=x1@entry=111, x2=x2@entry=-32766, format=0x437fdb "%s has gone %s.\n")
    at sendsys.c:205
#3  0x000000000041a2ea in movemob (x=x@entry=111) at mobile.c:739
#4  0x000000000041a482 in consid_move (mon=mon@entry=111) at mobile.c:647
#5  0x000000000041a5af in move_mobiles () at mobile.c:527
#6  0x0000000000428562 in on_timer () at timing.c:538
#7  0x000000000041745a in main_loop (m_socket=m_socket@entry=0) at main.c:544
#8  0x0000000000418312 in xmain (fd=<optimized out>) at main.c:172
#9  main (argc=<optimized out>, argv=<optimized out>, ep=<optimized out>) at main.c:90
(gdb)
```

albertw@sionnach:~/projects/dirt31/src — gdb -q --args ../bin/aberd -f

~/projects/dirt31/src

```
#4  0x000000000041a482 in consid_move (mon=mon@entry=111) at mobile.c:647
#5  0x000000000041a5af in move_mobiles () at mobile.c:527
#6  0x0000000000428562 in on_timer () at timing.c:538
#7  0x000000000041745a in main_loop (m_socket=m_socket@entry=0) at main.c:544
#8  0x0000000000418312 in xmain (fd=<optimized out>) at main.c:172
#9  main (argc=<optimized out>, argv=<optimized out>, ep=<optimized out>) at main.c:90
(gdb) list
167
168     char *check_send_msg(int plx, int a, char *t)
169     {
170       struct _send_msg_box *b = (struct _send_msg_box *)a;
171
172       if (test_rcv(plx,b->mode,b->min,b->max,b->x1,b->x2)) return t;
173       return NULL;
174     }
175
176     #ifdef VARGS
(gdb) print plx
$1 = 111
(gdb) print a
$2 = -13792
(gdb) print b->mode
Cannot access memory at address 0xfffffffffffca20
(gdb)
```

albertw@sionnach:~/projects/dirt31/src – vim sendsys.c

~/projects/dirt31/src

```
160
161 struct _send_msg_box {
162   int mode;
163   int min;
164   int max;
165   int x1;
166   int x2;
167 };
168
169 char *check_send_msg(int plx, intptr_t a, char *t)
170 {
171   struct _send_msg_box *b = (struct _send_msg_box *)a;
172
173   if (test_rcv(plx,b->mode,b->min,b->max,b->x1,b->x2)) return t;
174   return NULL;
175 }
176
177 #ifdef VARGS
178
179 void send_msg(int destination, /* Where to send to */
180           int mode,        /* Flags to control sending */
181           int min,         /* Minimum level of recipient */
182           int max,         /* Maximum level of recipient */
```

165,1 59%

>_ 13

A vanished preprocessor feature

original build command

```
sprintf(buff,
  "/lib/cpp -P -R -I../include %s",
  fn
);
```

```
#include "undef.h"
#include "cflags.h"
#include "exits.h"
%mobiles

Name            = bear
PName           = "The Bear"
Location        = Glen
```

The -R option allowed recursive macros. ANSI C removed that behaviour, but DIRT's zone files still depend on it. A modern compiler therefore cannot simply preprocess the world in the original way.

RESTORATION CHOICE

Write a small compatible preprocessor so the historical zone files remain usable.

(Ok, have claude write a small compatible preprocessor...)

Why not convert everything to JSON?

JSON REWRITE

- Easier for a new program to consume.
- Familiar tooling and schemas.
- Need to convert any new files
- Converted files become a new interpretation.

COMPATIBILITY

- Keep the original zone files authoritative.
- Isolate unusual preprocessing and parsing in a small, documented layer.
- The modern program adapts to the historical data, not the other way around.
- Some game mechanics are implemented in C and tied to the game worlds being compiled in.

It works!!! What next?

There may be more people interested in how early online games worked than people willing to learn early-1990s C in order to study one.

RESTORATION

Preserve the original implementation and make it runnable.

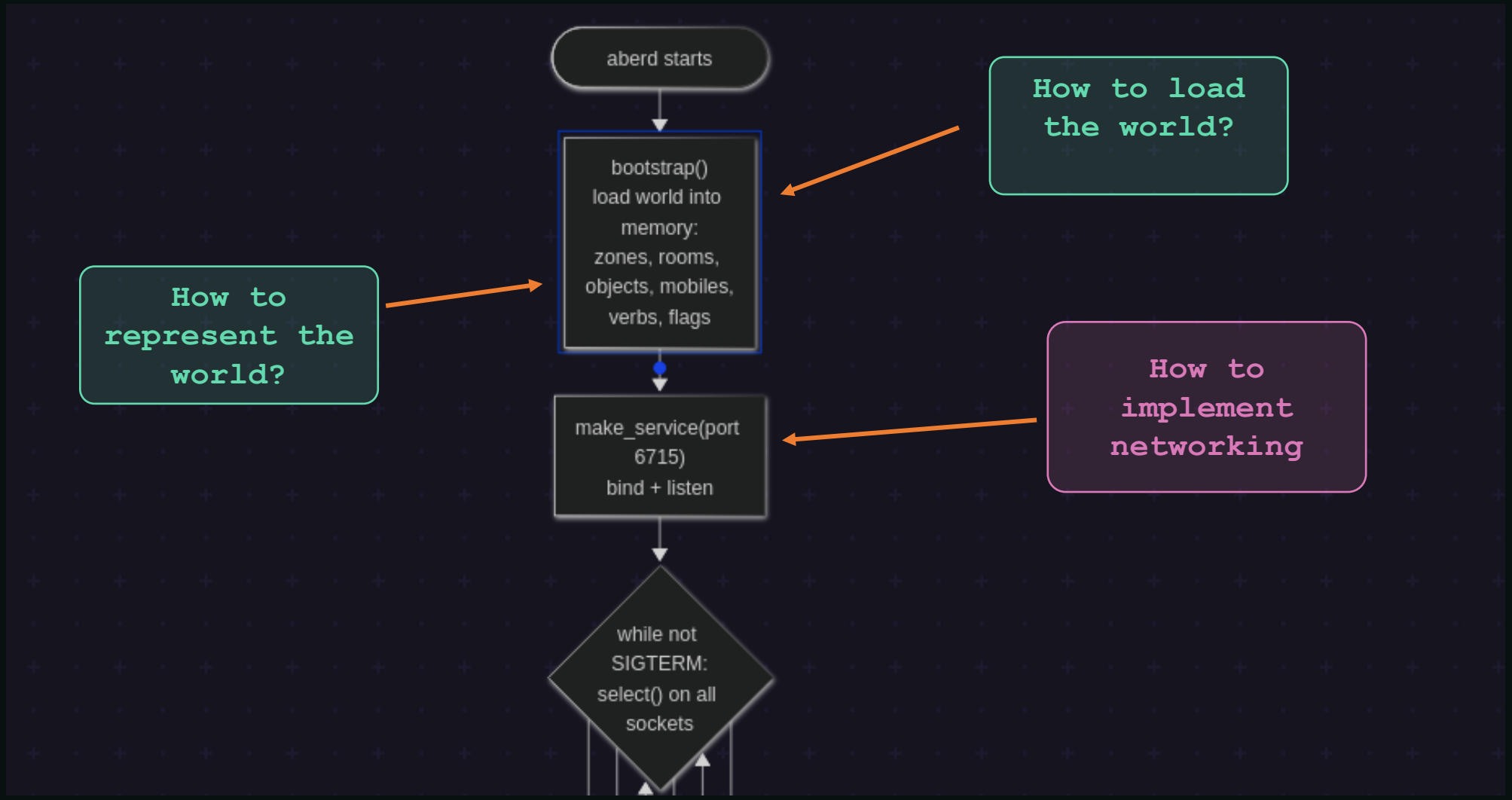


INTERPRETATION

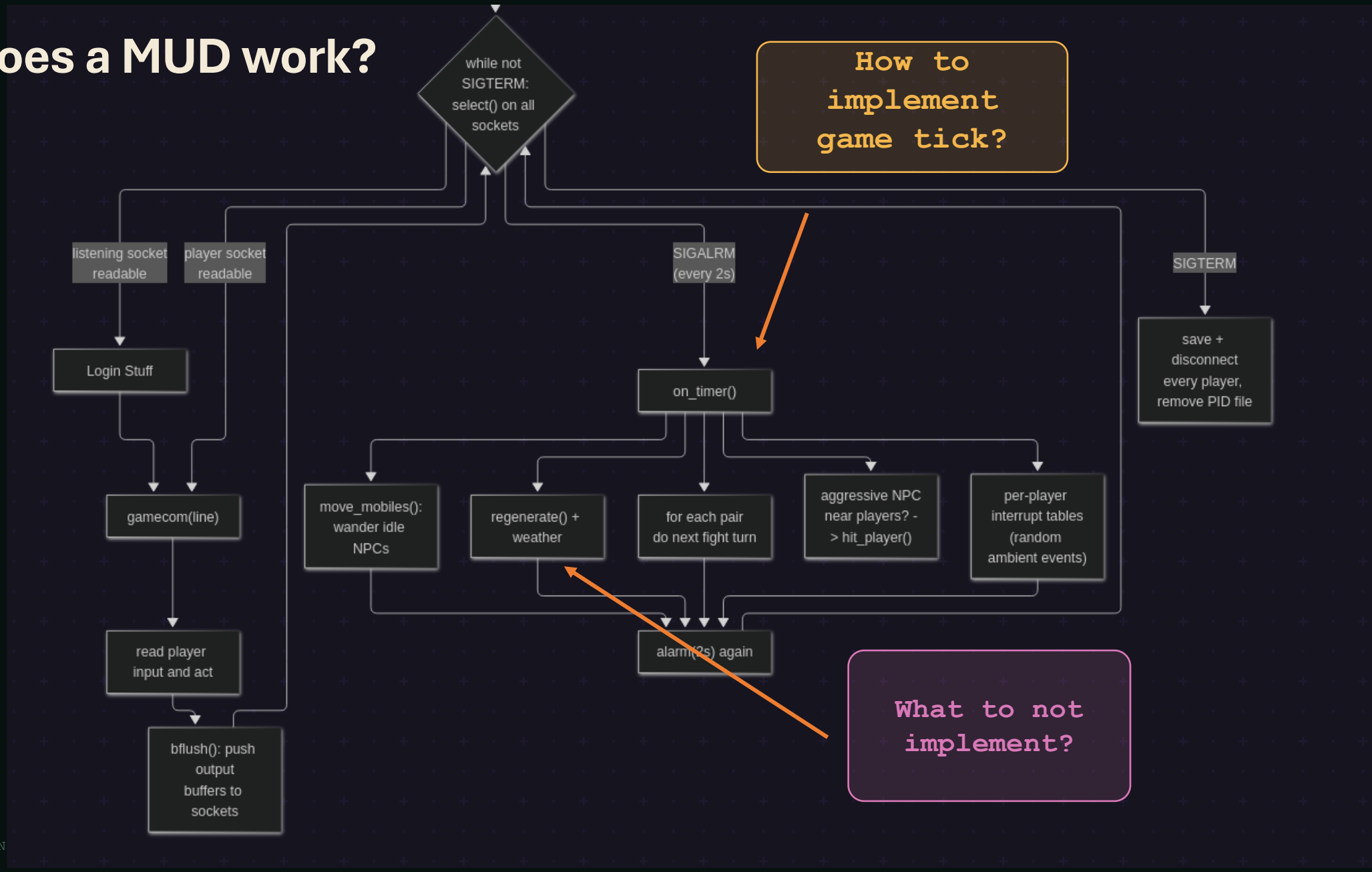
Preserve design and behaviour but express the implementation in modern Python.

The Python port is not a replacement for the C. It is a second artefact with a different educational purpose.

Preserving the game flow: How does a MUD work?



How does a MUD work?



Python implementation choices.

DATA TYPES	C implementation uses structs and arrays. Use Classes and Objects?
THREADS	Possible, but the old design deliberately avoids one thread per connection.
TWISTED	A capable networking framework, but adds a separate framework and conventions.
ASYNCIO	Standard Python, single-threaded event loop, explicit waiting and a close conceptual fit.

The AI disclaimer

Yes AI was used extensively in creating the port.

‘Hey Claude, turn this C code into python’ gets you nonworking gibberish.

PLAN

- Details on what to preserve and how.
- What are acceptable changes
- What our porting philosophy is.
- What our audience is.
- What the goal of our port is.
- What we want in scope and what we don’t as it will complicate the implementation.

IMPLEMENTATION

- Lots of testing and tweaking.
- In particular:
  - Telnet IAC (Interpret as command bytes)
  - Command processing

Asyncio 101

An async function can pause at await and give control back to the event loop. While one player waits for input or a timer, another player's coroutine can run.

three_players.py

```python
import asyncio

async def player(name):
    print(f"{name}: waiting")
    await asyncio.sleep(2)
    print(f"{name}: ready")

await asyncio.gather(
    player("Alice"), player("Bob"), player("Charlie")
)
```

SEQUENTIAL

2 + 2 + 2 = 6 seconds

ASYNC WAITING

about 2 seconds total

A small asyncio MUD server

server.py

```python
async def handle_player(reader, writer):
    writer.write(b"Welcome!\n> ")
    await writer.drain()

    while data := await reader.readline():
        command = data.decode().strip()
        await dispatch(command, writer)

async def game_tick():
    while True:
        await asyncio.sleep(2)
        update_world()

server = await asyncio.start_server(
    handle_player, "127.0.0.1", 4000
)
```

handle_player() waits for one connection without blocking everyone else.

game_tick() reproduces the periodic two-second rhythm used for world activity.

Both remain inside one event loop, preserving the original single-threaded mental model.

Parsing the zone files

C

- World is compiled into the program
- Players, mobiles are in one array
- Rooms are + integers, mobiles are –
- Objects have CARRIED_BY

```
Location *room_data;

Typedef struct{
    char *name;
    int exits[NEXITS];
    int_set objects;
    …
} Location
```

PYTHON

- World is parsed at runtime
- Rooms an objects are named
- Easier to follow and debug
- Mobiles carry Objects
- Room is now zone:name

```
@dataclass
Class Room:
    id: str
    exits: dict
    items:
    characters:
    …
```


>_

Other Changes

PLAYER DATA

Saved as per player json files. Easier to debug.
C used a single binary file.

COMMAND PARSER

Easier to handle string parsing in python than C.

WORLD RULES

Combat, movement, weather, inventory and permissions retain C game logic. (Where implemented!)

ENOTIMPLEMENTED

Weather / daylight
Wizard Commands
Special features (e.g. locked doors)
Spells

The future

DIRT

The audience is likely to be small: people interested in nostalgia, history and how early online worlds worked.

RECOVERY

More patches or zone files may still exist on old disks, archived university servers or forgotten FTP mirrors.

AI

LLMs may help explain code, create tests or prototype quests, but generated work needs human review and clear provenance. Mud worlds are very context dependent.

NEW MUDS

For a new Python MUD, Evennia (www.evennia.com/) is a stronger starting point because it is an active framework and community.

References and further reading

SOURCE	https://github.com/albertw/dirt https://github.com/albertw/pydirt
Archive.org archive	https://github.com/albertw/NLwebsite
VIRTUAL WORLDS	Richard Bartle. Designing Virtual Worlds. mud.co.uk/richard/DesigningVirtualWorlds.pdf
NEW PYTHON MUDS	Evennia, an active Python framework for building multiplayer text worlds. www.evennia.com
ME	https://www.linkedin.com/in/albertw

Questions? >_