Bringing a jewel-encrusted warhammer to a knife fight

Mike Seddon

The beforetimes ....

1995

2004

2007

But also in 1995 …

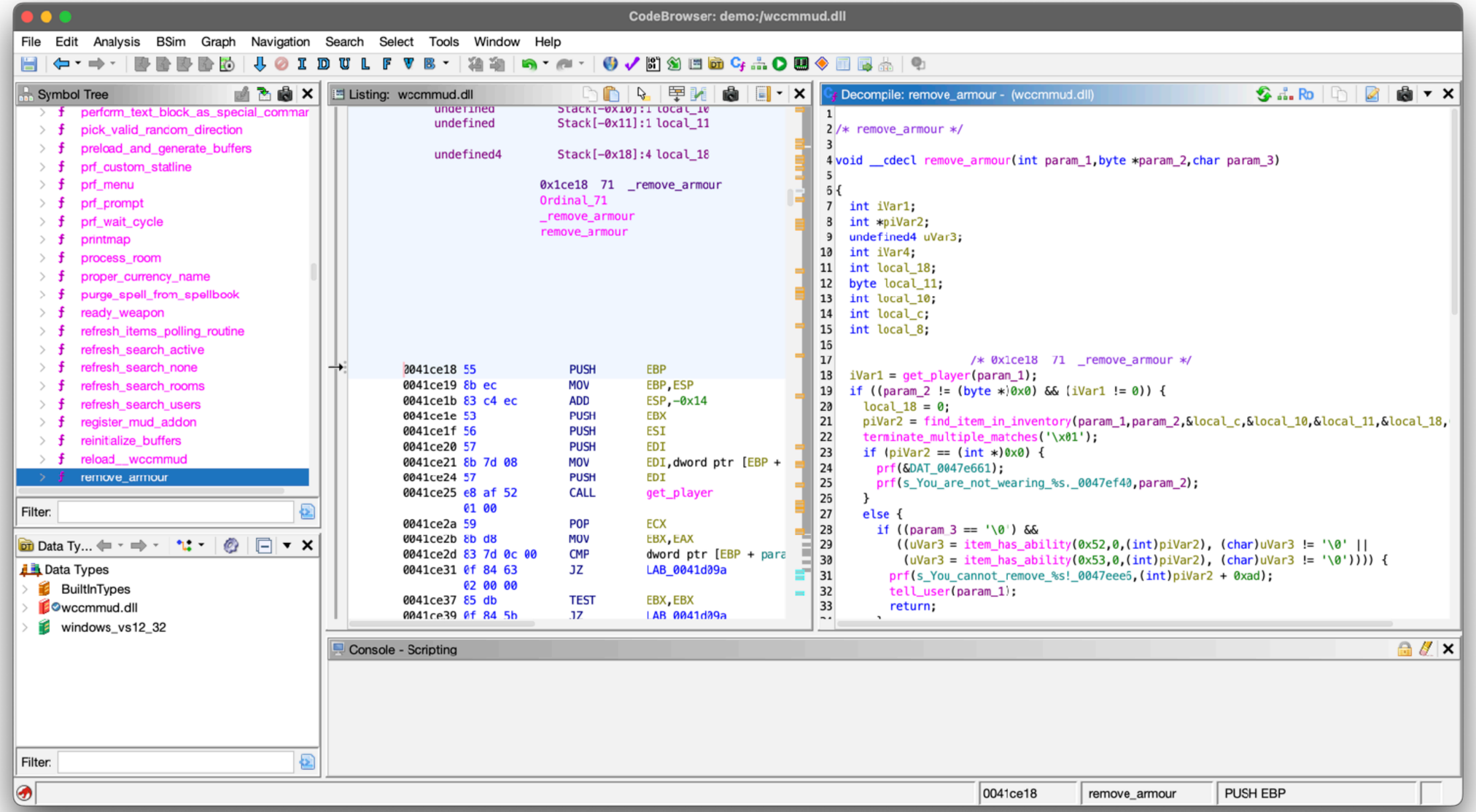
Bulletin Boards and MUDs?!

```
gravestones popping out of the mist around you, and you can barely see what
appears to be a tomb just a little ways ahead. Paths lead to the north and
east, and the bridge that connects the graveyard to the town is to the
southwest.
You notice 3 silver nobles, 1 copper farthing, gravestone here.
Also here: gravedigger.
Obvious exits: north, east, closed door west, southwest
A zombie shambles into the room from the east.
The zombie swings at you with its arm!
[HP=112/MA=28]:
Graveyard, Entry
You notice 3 silver nobles, 1 copper farthing, gravestone here.
Also here: gravedigger, zombie.
Obvious exits: north, east, closed door west, southwest
[HP=112/MA=28]:a zo
*Combat Engaged*
You impale zombie for 8 damage!
You impale zombie for 15 damage!
The zombie swings at you with its arm!
The zombie swings at you with its arm!
You impale zombie for 6 damage!
You pierce zombie for 17 damage!
You pierce zombie for 15 damage!
[HP=112/MA=28]:
```

Can we Rewrite it in Rust

… without the original source and just a DLL?

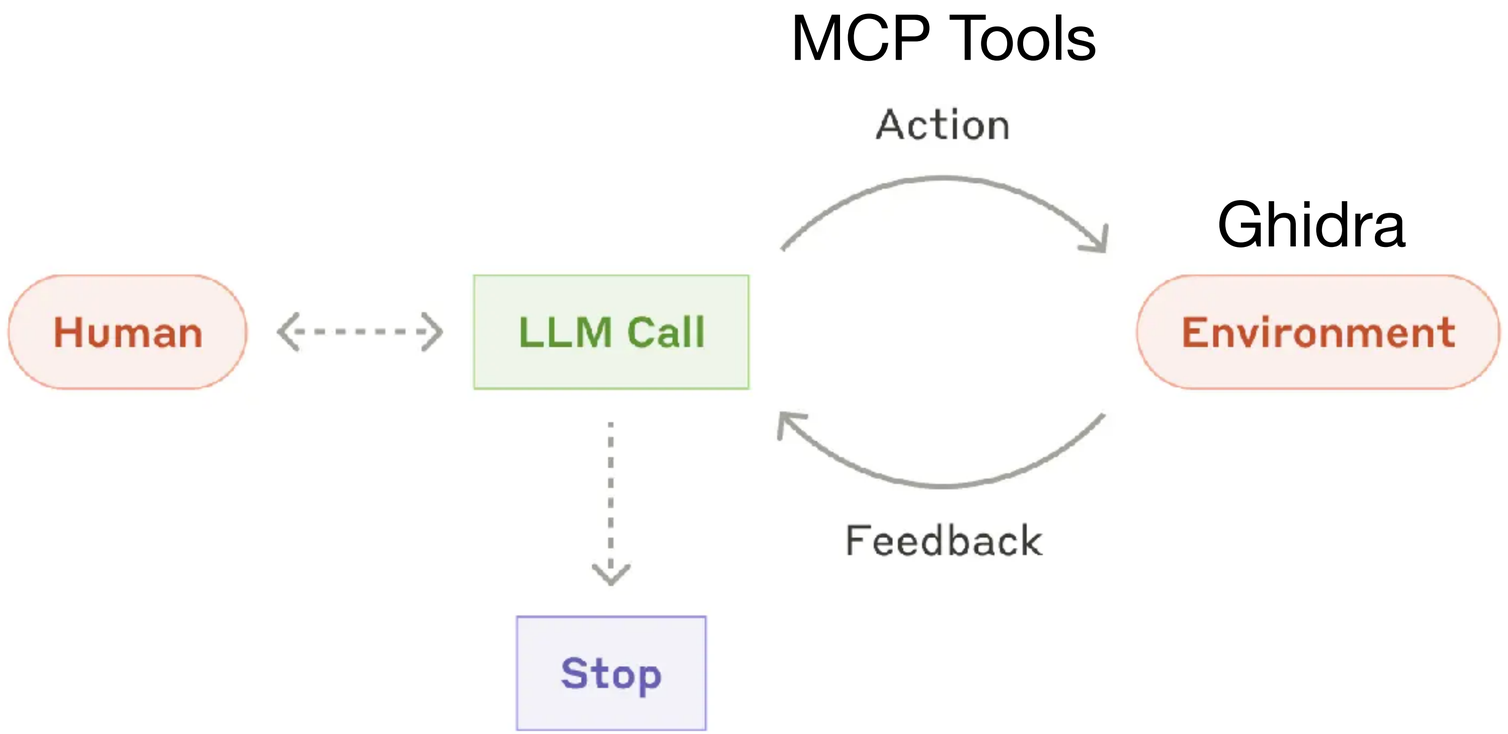
Ghidra Decompiler



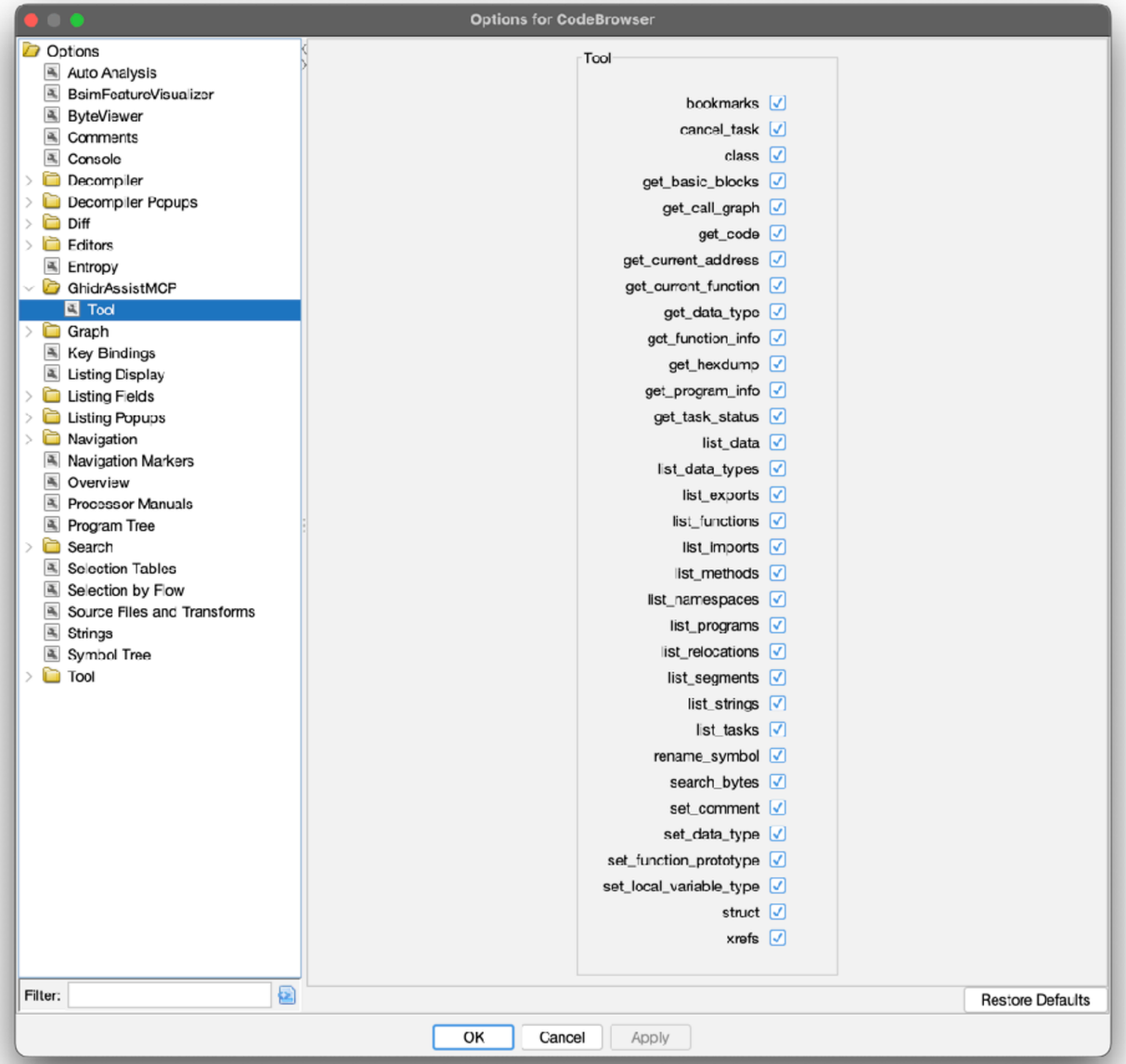
Obfuscated pseudo-C is good … but…

```
/* monster_add_cast_spell_to_user */

int __cdecl
monster_add_cast_spell_to_user
          (int param_1,uint param_2,undefined2 param_3,int param_4,int param_5,int param_6,
          char param_7)

{
  int iVar1;
  int iVar2;
  int iVar3;

                    /* 0x25ea6  166  _monster_add_cast_spell_to_user */
  iVar1 = get_player(param_1);
  if ((iVar1 != 0) && (param_5 != 0)) {
    iVar3 = -1;
    iVar2 = 0;
    do {
      if (*(ushort *)(iVar1 + 0x40 + iVar2 * 2) == param_2) {
        if ((param_7 != '\0') && (*(short *)(iVar1 + 0x54 + iVar2 * 2) < param_4)) {
          *(short *)(iVar1 + 0x54 + iVar2 * 2) = (short)param_4;
          *(undefined2 *)(iVar1 + 0x68 + iVar2 * 2) = param_3;
          monster_display_spell_success(param_1,param_5,param_6,iVar1 + 0x1e,param_4);
          return iVar2;
        }
        return -2;
      }
      if (*(short *)(iVar1 + 0x40 + iVar2 * 2) == 0) {
        iVar3 = iVar2;
      }
      iVar2 = iVar2 + 1;
    } while (iVar2 < 10);
    if (iVar3 != -1) {
      *(undefined2 *)(iVar1 + 0x40 + iVar3 * 2) = (undefined2)param_2;
      *(undefined2 *)(iVar1 + 0x68 + iVar3 * 2) = param_3;
      *(short *)(iVar1 + 0x54 + iVar3 * 2) = (short)param_4;
      monster_display_spell_success(param_1,param_5,param_6,iVar1 + 0x1e,param_4);
      return iVar3;
    }
  }
  return -1;
}
```

Gemini enters the game …



Equip Gemini with the correct tools



get_code: get the decompiled code

xrefs: find function callers/callees

rename_symbol: rename variables

Add time + money and …

```
/* Adds a cast spell effect to a player's active spell list. This function attempts to find an
   existing spell by `spell_id`. If found and `can_overwrite_if_stronger` is true, the spell's
   duration and value are updated if the new value is higher. Otherwise, it searches for an empty
   spell slot to add the new spell. A success message is displayed to the player upon successful
   spell addition or update. */

int __cdecl
monster_add_cast_spell_to_user
          (int player_id,ushort spell_id,short duration,int value,spell *spell_ptr,
          monster *monster_ptr,bool can_overwrite_if_stronger_flag)

{
  player *player_ptr;
  int spell_slot_index;
  int empty_spell_slot_index;
  undefined2 unused_stack_var;

                    /* 0x25ea6  166  _monster_add_cast_spell_to_user */
  player_ptr = get_player(player_id);
  if ((player_ptr != (player *)0x0) && (spell_ptr != (spell *)0x0)) {
    empty_spell_slot_index = -1;
    spell_slot_index = 0;
    do {
      if (player_ptr->spell_ids[spell_slot_index] == _spell_id) {
        if ((can_overwrite_if_stronger_flag) && (player_ptr->spell_values[spell_slot_index] < value)
           ) {
          player_ptr->spell_values[spell_slot_index] = (short)value;
          player_ptr->spell_durations[spell_slot_index] = duration;
          monster_display_spell_success
                    (player_id,spell_ptr,monster_ptr,player_ptr->given_name,(char *)value);
          return spell_slot_index;
        }
        return -2;
      }
      if (player_ptr->spell_ids[spell_slot_index] == 0) {
        empty_spell_slot_index = spell_slot_index;
      }
      spell_slot_index = spell_slot_index + 1;

...
```

Cool story, bro …

… but where is Ferris?

Claude enters the game …

```rust
/// Adds a spell cast by a monster to a player's active spells.
///
/// Handles checking if a spell already exists and optionally overwriting it
/// if the new value is stronger. Uses the last available empty slot.
///
/// Returns Some(slot_index) on success, or None on failure.
pub fn monster_add_cast_spell_to_user(
    &self,
    player_id: PlayerId,
    spell: &'static Spell,
    duration: i16,
    value: i16,
    can_overwrite_if_stronger: bool,
    game: &mut Game,
    messages: &'static Messages,
    players: &mut Players,
    rooms: &Rooms,
) -> Option<usize> {
    let player = players.get(&player_id)?;

    // Find existing spell or last empty slot
    let mut empty_slot = None;
    let mut existing_slot = None;

    for (i, active_spell_opt) in player.active_spells.iter().enumerate() {
        match active_spell_opt {
            Some(active_spell) if active_spell.spell.id == spell.id => {
                existing_slot = Some((i, active_spell.value));
                break;
            }
            None => empty_slot = Some(i),
            _ => {}
        }
    }

    // Determine which slot to use and whether to proceed
    let slot = match existing_slot {
        Some((i, existing_value)) => {
            if can_overwrite_if_stronger && existing_value < value {
                i
            } else {
                return None;
            }
        }
        None => empty_slot?,
    };

    // Get player name before mutable borrow, then update the spell slot
    let player_name = player.given_name.clone();
    players.get_mut(&player_id).unwrap().active_spells[slot] =
        Some(PlayerActiveSpell { spell, value, duration });

    self.monster_display_spell_success(
        Some(player_id),
        spell,
        &player_name,
        &value.to_string(),
        game,
        messages,
        players,
        rooms,
    );

    Some(slot)
}
```

The confident accountant

Interviewer: “I heard you are extremely quick at math.”

Me: “That’s correct.”

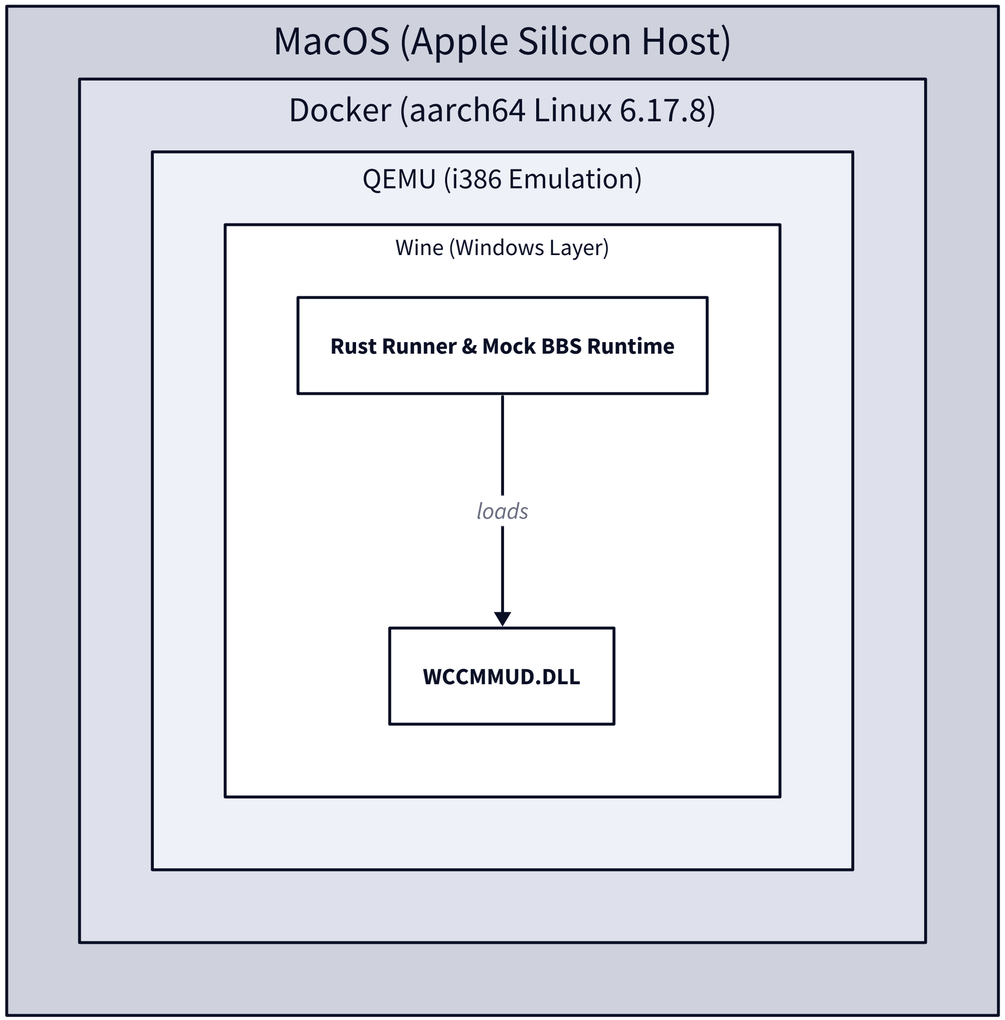
Interviewer: “What is 14 multiplied by 27?”

Me: “600!”

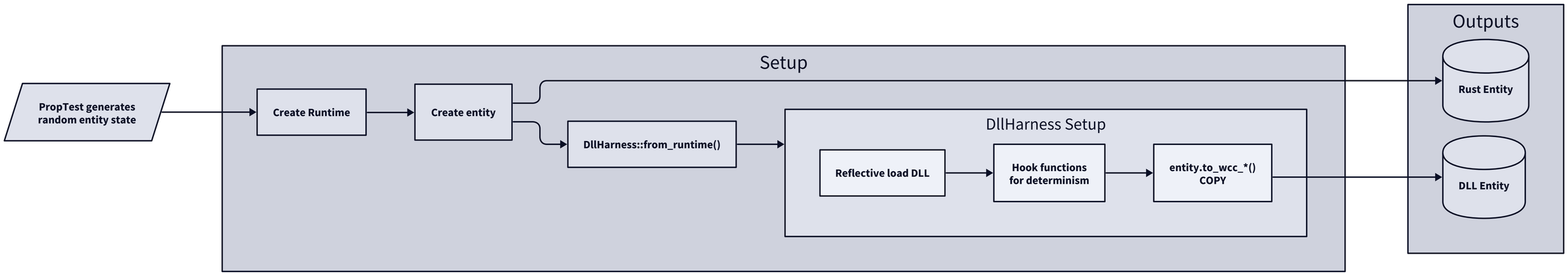
Interviewer: “That’s not even close.”

Me: “Yeah, but it was fast.”

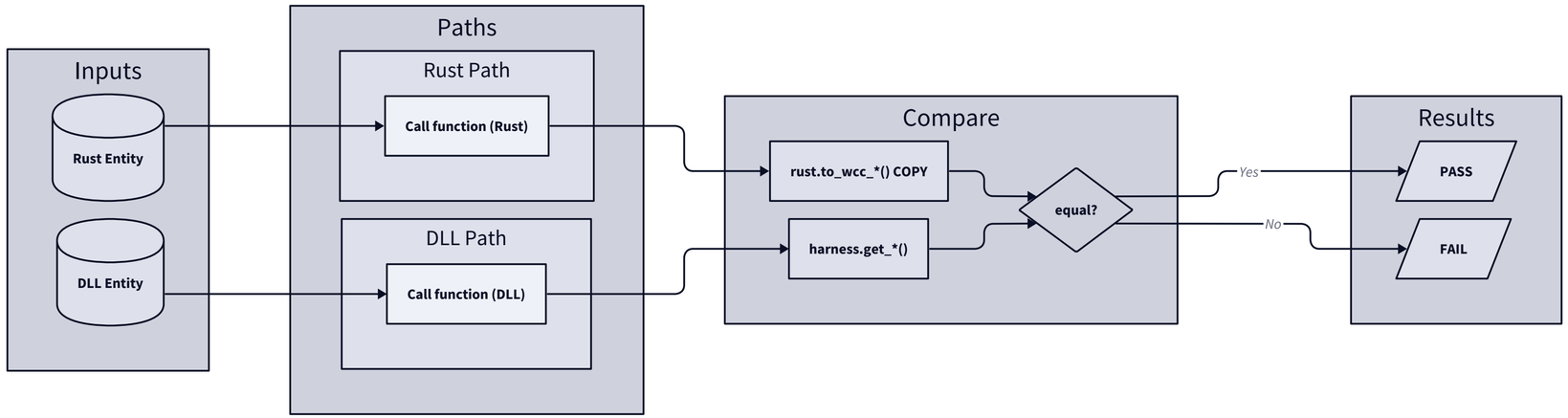
It’s QEMU all the way down …



Setup the state …



… and execute the tests



PropTest and deterministic simulation

```
proptest! {
    #![proptest_config(ProptestConfig::with_cases(256))]
    #[test]
    fn test_add_delay_differential(
        initial_delay in any::<u8>(),
        delay_to_add in any::<u8>(),
    ) {
        // Create the rust state
        let mut rt = get_test_runtime();

        // Create the player
        let mut player = Player::new(“PropTestPlayer".to_string(), &mut rt.game, &rt.races[&RaceType::Human], &rt.classes[&ClassType::Warrior]);

        // Mutate the player
        player.delay = initial_delay;

        // Create the Windows world from the Rust world
        #[cfg(target_os = "windows")]
        let mut harness = unsafe { crate::differential::dll_harness::DllHarness::from_runtime(&rt) };

        // Inject the player to match Rust
        #[cfg(target_os = “windows")]
        let harness_slot = {
            let slot = player.id.0 as u16;
            harness.state.players.insert(slot, Box::new(player.to_wcc_player()));
            slot
        };

        // Run Rust implementation
        player.add_delay(delay_to_add);

        // Windows: Run DLL and compare
        #[cfg(target_os = "windows")]
        {
            unsafe { harness.add_delay(harness_slot, delay_to_add); }
            let dll_player = unsafe { harness.get_player(harness_slot).unwrap() };
            prop_assert_eq!(player, dll_player);
        }
    }
}
```

Arbitrary is your friend

```rust
#[cfg(test)]
impl SpellId {
    /// Generate a random valid spell ID (hardcoded range from game database: 1..=1398).
    pub fn arbitrary() -> impl proptest::strategy::Strategy<Value = SpellId> {
        use proptest::prelude::*;
        (1u32..=1398).prop_map(SpellId)
    }
}
```

Other interesting learnings

1. The original game was deeply unsafe making the borrow checker interesting.

2. The original struct byte arrays have a lot of garbage bytes making roundtrip difficult.

3. Change detection is a hard problem which was an interesting side quest: including a custom rustc.

Links

Blog:
https://reorchestrate.com/posts/bringing-a-warhammer-to-a-knife-fight/

GhidrAssistMCP
https://github.com/jtang613/GhidrAssistMCP

Ferris 3D
https://github.com/RayMarch/ferris3d