Newbie Shell Design Document - Implementation Update

Current Status (Implementation Phase)

Architectural Decisions Implemented

1. Universal & Prefix System

BREAKING CHANGE FROM ORIGINAL DESIGN: All commands now require `&` prefix for consistency.

New Syntax:

- `&find pattern &in file` → silent execution, no output
- `&show &find pattern &in file` → displays results with formatting
- `&show filename` → displays file contents
- `&exit` → quit shell

Benefits:

- Eliminates parsing ambiguity between commands and filenames
- Single rule: `&` = parsing instructions, no `&` = literal data
- Architecturally simpler: uniform token processing
- No special cases or context-dependent rules

2. Explicit Output Control (Implemented)

- Commands execute silently by default (scriptability)
- Only `&show` command produces visible output
- `&raw` modifier only follows `&show` on same line
- `&find` returns data structures, `&show` handles display

3. Pattern Language Status

Core Issue Identified: The `&start...&end` pattern matching has complex streaming state machine bugs.

Current Pattern Syntax (Needs Refinement):

- `&start use &letters &end` → should match lines beginning with "use" followed by letters
- **SYNTAX ISSUE**: Should be `&start=use` per original design document
- Current implementation has literal matching bugs in streaming architecture

Implementation Strategy Change

Standalone Pattern Matching Tool

Decision: Extract pattern matching into separate `~/find` project to isolate complexity.

Rationale:

- Shell architecture is sound but pattern matching engine needs focused development
- Streaming state machine debugging is complex when mixed with command parsing
- Need to validate natural language pattern concept before full integration

Development Plan:

1. Build standalone `find` tool with simple line-by-line processing
2. Perfect `&start...&end` pattern matching without streaming complexity
3. Add comprehensive test cases for pattern language
4. Integrate proven pattern engine back into main shell

Technical Debt Identified

Pattern Matching Bugs:

- Literal matching fails during character-by-character streaming
- Anchor logic (`&start`/`&end`) not properly enforcing boundaries
- State machine resets incorrectly during match failures
- Matches lines that don't contain required literal text

Complexity Management:

- Multiple interacting state machines (parsing, matching, anchoring, streaming)
- Edge cases compound exponentially in current architecture
- Need isolated testing environment for pattern engine

Working Components

Shell Architecture (Stable)

- Universal `&` prefix parsing ✓
- Command dispatching ✓
- Silent execution with explicit output control ✓

- `show` command with modifiers ✓

Pattern Language (Under Development)

- Pattern parsing logic works for simple cases
- Streaming implementation has state management bugs
- Need to implement `&start=value` syntax from original design
- Anchor enforcement needs debugging

Next Steps

1. **Complete standalone `find` tool** (Priority 1)
   - Implement pattern parsing for `&start`, `&end`, `&letters`, `&numbers`
   - Use simple line processing, not streaming
   - Add comprehensive test cases
   - Validate natural language approach vs regex
2. **Pattern Syntax Refinement** (Priority 2)
   - Implement `&start=value` and `&end=value` syntax
   - Add proximity matching (`&within N words of`)
   - Add logic operators (`&or`, `&not`, `&maybe`)
3. **Integration** (Priority 3)
   - Integrate proven pattern engine back into main shell
   - Add streaming architecture after basic functionality works
   - Implement three-thread architecture if performance requires it

Design Philosophy Validation

Confirmed Principles:

- Delimiter-free parsing works with universal `&` prefix
- Explicit output control enables scriptability
- Natural language commands reduce cognitive load

Under Validation:

- Whether `&start use &letters &end` is actually more readable than `^use[a-zA-Z]*$`
- Performance characteristics of streaming vs traditional approaches

- User adoption of natural language pattern syntax

Development Constraints

Daily Work Capacity: ~5 hours of deep technical work at this complexity level **Current Focus**: Standalone pattern matching tool to isolate complexity **Complexity Management**: Break down problems rather than debugging multiple interacting systems

This represents the current state after implementing core shell architecture and identifying the pattern matching engine as the primary remaining challenge.