Clara's Cerebrum - Project Layout

Overview

This document defines the file and directory structure for the Clara's Cerebrum CLIPS REST API service, organized to support clean architecture, testability, and evolution from subprocess to FFI-based integration.

Root Directory Structure

```
clara-cerebrum/
├── Cargo.toml                 # Workspace root manifest
├── Cargo.lock                 # Dependency lockfile
├── README.md                  # Project overview and quick start
├── LICENSE                    # License file
├── .gitignore                 # Git ignore patterns
├── .env.example               # Example environment variables
├── rust-toolchain.toml        # Rust version pinning
├── deny.toml                  # Cargo-deny configuration for supply chain security
├── clippy.toml                # Clippy lints configuration
├── rustfmt.toml               # Code formatting rules
│
├── docs/                      # Documentation
│   ├── ADR/                   # Architecture Decision Records
│   │   ├── 001-backend-selection.md
│   │   ├── 002-persistence-format.md
│   │   └── 003-security-model.md
│   ├── API.md                 # API specification and examples
│   ├── DEPLOYMENT.md          # Deployment guide
│   ├── SECURITY.md            # Security practices and threat model
│   └── DEVELOPMENT.md         # Development setup and guidelines
│
├── config/                    # Configuration files
│   ├── default.toml           # Default configuration
│   ├── development.toml       # Development overrides
│   ├── production.toml        # Production settings
│   └── schema.json            # Configuration schema for validation
│
├── scripts/                   # Utility scripts
│   ├── setup-dev.sh           # Development environment setup
│   ├── run-tests.sh           # Test runner with coverage
│   ├── benchmark.sh           # Performance benchmarking
│   ├── security-audit.sh      # Security scanning
│   └── docker-build.sh        # Container build script
│
├── migrations/                # Database migrations (if using DB for persistence)
│   └── 001_initial_schema.sql
│
├── clips/                     # CLIPS-related assets
│   ├── binaries/              # CLIPS executables (gitignored, downloaded)
│   │   └── .gitkeep
│   ├── rules/                 # Baseline rule libraries
│   │   ├── base_rules.clp
│   │   └── examples/
│   └── tests/                 # CLIPS test scripts
│       ├── basic.clp
│       ├── persistence.clp
│       └── stress.clp
│
```

```
├── crates/                    # Rust workspace crates
│   ├── clara-api/             # REST API server
│   ├── clara-core/            # Core business logic
│   ├── clara-clips/           # CLIPS integration layer
│   ├── clara-session/         # Session management
│   ├── clara-persistence/     # Persistence and storage
│   ├── clara-security/        # Security and auth
│   ├── clara-metrics/         # Observability and metrics
│   └── clara-config/          # Configuration management
│
├── tests/                     # Integration tests
│   ├── integration/
│   │   ├── eval_tests.rs
│   │   ├── session_tests.rs
│   │   ├── persistence_tests.rs
│   │   └── security_tests.rs
│   ├── load/
│   │   ├── concurrent_eval.rs
│   │   └── session_churn.rs
│   ├── chaos/
│   │   ├── timeout_storms.rs
│   │   └── process_kill.rs
│   └── fixtures/
│       ├── test_rules.clp
│       └── test_data.json
│
├── benches/                   # Benchmark suites
│   ├── eval_throughput.rs
│   └── session_lifecycle.rs
│
├── docker/                    # Container definitions
│   ├── Dockerfile             # Production image
│   ├── Dockerfile.dev         # Development image
│   └── docker-compose.yml     # Local stack
│
└── .github/                   # GitHub-specific files
    └── workflows/
        ├── ci.yml             # Continuous integration
        ├── security.yml       # Security scanning
        └── release.yml        # Release automation
```

Detailed Crate Structure

1. clara-api/

The HTTP server and API endpoint handlers.

```
clara-api/
├── Cargo.toml
├── src/
│   ├── main.rs                # Binary entry point
│   ├── lib.rs                 # Library exports
│   ├── server.rs              # Axum server setup
│   ├── middleware/
│   │   ├── mod.rs
│   │   ├── auth.rs            # JWT/token validation
│   │   ├── rate_limit.rs      # Rate limiting middleware
│   │   ├── cors.rs            # CORS configuration
│   │   └── tracing.rs         # Request tracing
│   ├── routes/
│   │   ├── mod.rs
│   │   ├── eval.rs            # POST /eval
│   │   ├── sessions.rs        # Session CRUD endpoints
│   │   ├── health.rs          # Health check endpoints
│   │   └── metrics.rs         # Metrics endpoint
│   ├── handlers/
│   │   ├── mod.rs
│   │   ├── eval_handler.rs    # Ephemeral eval logic
│   │   ├── session_handler.rs # Persistent session handlers
│   │   └── error_handler.rs   # Error response formatting
│   ├── models/
│   │   ├── mod.rs
│   │   ├── request.rs         # API request types
│   │   ├── response.rs        # API response types
│   │   └── error.rs           # Error types
│   └── validation/
│       ├── mod.rs
│       └── input.rs           # Request validation logic
└── tests/
    └── api_tests.rs
```

2. clara-core/

Core business logic independent of transport layer.

```
clara-core/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── service/
│   │   ├── mod.rs
│   │   ├── eval_service.rs        # Evaluation orchestration
│   │   ├── session_service.rs     # Session lifecycle management
│   │   └── load_service.rs        # Rule/fact loading
│   ├── types/
│   │   ├── mod.rs
│   │   ├── session.rs             # Session metadata
│   │   ├── eval_result.rs         # Evaluation results
│   │   └── resource_limits.rs     # Quota definitions
│   ├── error.rs                   # Domain error types
│   └── traits.rs                  # Core trait definitions
└── tests/
    └── service_tests.rs
```

3. clara-clips/

CLIPS integration layer with subprocess and future FFI backends.

```
clara-clips/
├── Cargo.toml
├── build.rs                       # Build script for FFI bindings
├── src/
│   ├── lib.rs
│   ├── backend/
│   │   ├── mod.rs
│   │   ├── subprocess/
│   │   │   ├── mod.rs
│   │   │   ├── process.rs         # Subprocess spawn/management
│   │   │   ├── protocol.rs        # REPL protocol (sentinel, framing)
│   │   │   └── io.rs              # Stdin/stdout handling
│   │   └── ffi/                   # Future FFI implementation
│   │       ├── mod.rs
│   │       ├── bindings.rs        # Generated FFI bindings
│   │       ├── environment.rs     # FFI environment management
│   │       └── conversion.rs      # Data type conversion
│   ├── executor.rs                # Unified executor trait
│   ├── command.rs                 # Command abstraction
│   ├── output.rs                  # Output parsing and normalization
│   ├── error.rs                   # CLIPS-specific errors
│   └── timeout.rs                 # Timeout enforcement
└── tests/
    ├── subprocess_tests.rs
    └── protocol_tests.rs
```

4. clara-session/

Session management and lifecycle.

```
clara-session/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── manager.rs              # Session manager (create, lookup, evict)
│   ├── store.rs                # In-memory session store
│   ├── lifecycle.rs            # Session lifecycle FSM
│   ├── eviction.rs             # LRU eviction policy
│   ├── queue.rs                # Per-session eval queue
│   └── metadata.rs             # Session metadata tracking
└── tests/
    ├── manager_tests.rs
    └── eviction_tests.rs
```

5. clara-persistence/

Session state persistence and reload.

```
clara-persistence/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── format/
│   │   ├── mod.rs
│   │   ├── json.rs             # JSON serialization format
│   │   └── binary.rs           # Future: binary format
│   ├── storage/
│   │   ├── mod.rs
│   │   ├── filesystem.rs       # File-based storage
│   │   └── s3.rs               # Future: S3 backend
│   ├── codec.rs                # Compression/encryption
│   ├── integrity.rs            # Checksum verification
│   └── migration.rs            # Format version migration
└── tests/
    └── roundtrip_tests.rs
```

6. clara-security/

Security, authentication, and authorization.

```
clara-security/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── auth/
│   │   ├── mod.rs
│   │   ├── jwt.rs              # JWT token validation
│   │   ├── rbac.rs             # Role-based access control
│   │   └── scopes.rs           # Permission scopes
│   ├── filter/
│   │   ├── mod.rs
│   │   ├── command_filter.rs   # CLIPS command deny/allow-list
│   │   └── path_validator.rs   # File path validation
│   ├── sandbox/
│   │   ├── mod.rs
│   │   ├── limits.rs           # Resource limits (CPU, memory)
│   │   └── isolation.rs        # Process isolation helpers
│   ├── audit.rs                # Audit logging
│   └── crypto.rs               # Encryption utilities
└── tests/
    ├── filter_tests.rs
    └── auth_tests.rs
```

7. clara-metrics/

Observability: metrics, tracing, and logging.

```
clara-metrics/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── metrics/
│   │   ├── mod.rs
│   │   ├── counters.rs         # Counter metrics
│   │   ├── histograms.rs       # Latency histograms
│   │   └── gauges.rs           # Gauge metrics
│   ├── tracing/
│   │   ├── mod.rs
│   │   ├── setup.rs            # Tracing subscriber setup
│   │   └── spans.rs            # Custom span utilities
│   ├── logging/
│   │   ├── mod.rs
│   │   ├── structured.rs       # Structured log formatting
│   │   └── redaction.rs        # Sensitive data redaction
│   └── exporter.rs             # Prometheus/OTLP exporters
└── tests/
    └── metrics_tests.rs
```

8. clara-config/

Configuration management and validation.

```
clara-config/
├── Cargo.toml
├── src/
│   ├── lib.rs
│   ├── loader.rs              # Config file loading
│   ├── schema.rs              # Config schema definition
│   ├── validation.rs          # Config validation
│   ├── defaults.rs            # Default values
│   └── env.rs                 # Environment variable overrides
└── tests/
    └── config_tests.rs
```

Testing Structure

Integration Tests Directory

```
tests/integration/
├── common/
│   ├── mod.rs                 # Shared test utilities
│   ├── fixtures.rs            # Test fixture management
│   ├── harness.rs             # Test harness setup
│   └── assertions.rs          # Custom assertions
├── eval_tests.rs              # POST /eval endpoint tests
├── session_tests.rs           # Session lifecycle tests
├── persistence_tests.rs       # Save/reload tests
├── security_tests.rs          # Auth and filtering tests
├── concurrency_tests.rs       # Concurrent access tests
└── api_contract_tests.rs      # API schema validation
```

Load Tests

```
tests/load/
├── scenarios/
│   ├── burst_eval.rs          # Burst evaluation load
│   ├── sustained_sessions.rs  # Long-lived sessions
│   └── mixed_workload.rs      # Mixed ephemeral/persistent
└── README.md                  # Load test documentation
```

Chaos Tests

```
tests/chaos/
├── timeout_storms.rs          # Timeout cascades
├── process_kill.rs            # Subprocess termination
├── resource_exhaustion.rs     # Memory/CPU limits
└── network_partition.rs       # Future: distributed scenarios
```

Configuration Examples

config/default.toml

toml

```toml
[server]
host = "127.0.0.1"
port = 8080
request_timeout_ms = 30000
max_request_body_size = 1048576  # 1MB

[clips]
binary_path = "./clips/binaries/clips"
handshake_timeout_ms = 5000
default_eval_timeout_ms = 2000
sentinel_marker = "__END__"

[sessions]
max_concurrent = 100
max_per_user = 10
eviction_policy = "lru"
default_ttl_seconds = 3600

[resources]
max_facts_per_session = 1000
max_rules_per_session = 500
max_memory_mb = 128
max_eval_queue_depth = 10

[security]
deny_list = ["system", "load", "save", "open", "close"]
allow_list_mode = false
allowed_file_paths = ["./clips/rules"]

[persistence]
enabled = false
storage_backend = "filesystem"
storage_path = "./data/sessions"
compression = "gzip"
encryption = true

[observability]
log_level = "info"
metrics_enabled = true
metrics_port = 9090
tracing_enabled = true
tracing_endpoint = "http://localhost:4317"

[auth]
jwt_secret = "${JWT_SECRET}"
token_expiry_seconds = 3600
```

Docker Structure

Dockerfile (Production)

dockerfile

```
FROM rust:1.82-slim as builder
WORKDIR /build
COPY . .
RUN cargo build --release

FROM debian:bookworm-slim
RUN apt-get update && apt-get install -y \
    ca-certificates \
    && rm -rf /var/lib/apt/lists/*
COPY --from=builder /build/target/release/clara-api /usr/local/bin/
COPY ./clips/binaries/clips /usr/local/bin/
COPY ./config /etc/clara-cerebrum/
EXPOSE 8080 9090
CMD ["clara-api"]
```

CI/CD Workflow Structure

.github/workflows/ci.yml

Key stages:

- Lint (clippy, rustfmt)
- Test (unit, integration, load subset)
- Security (cargo-audit, cargo-deny)
- Build (debug and release)
- Coverage (tarpaulin or llvm-cov)

Key Design Principles

1. Separation of Concerns

- API layer handles HTTP/routing
- Core contains business logic
- CLIPS layer abstracts backend
- Security is a separate concern

2. Testability

- Each crate has its own test suite
- Integration tests use fixtures
- Load/chaos tests isolated

3. Configurability

- TOML-based configuration
- Environment variable overrides
- Validation at startup

4. Evolvability

- Backend abstraction supports subprocess→FFI migration
- Versioned persistence format
- ADRs document major decisions

5. Observability

- Structured logging throughout
- Metrics in dedicated crate
- Distributed tracing support

6. Security by Default

- Security crate centralizes concerns
- Deny-list enforced
- Audit logging built-in

Development Workflow

Initial Setup

```bash
./scripts/setup-dev.sh          # Install dependencies, download CLIPS
cargo build                     # Build all crates
cargo test                      # Run test suite
```

Running Locally

```bash
export CLIPS_BINARY_PATH=./clips/binaries/clips
cargo run --bin clara-api       # Start API server
```

Running Tests

```bash
./scripts/run-tests.sh          # Unit + integration tests with coverage
cargo test --test load          # Load tests
cargo test --test chaos         # Chaos tests
```

Adding a New Feature

1. Create feature branch
2. Implement in appropriate crate(s)
3. Add unit tests
4. Add integration test if needed
5. Update API.md if endpoint changed
6. Submit PR

Migration Path: Subprocess → FFI

Phase 1: Subprocess (Current)

- Implement all crates with subprocess backend
- Stabilize API and lifecycle
- Achieve test coverage targets

Phase 2: FFI Preparation

- Build FFI bindings in `clara-clips/src/backend/ffi/`
- Add feature flag `ffi-backend` to Cargo.toml
- Implement executor trait for FFI backend

Phase 3: Dual-Stack

- Support both backends via config: `clips.backend = "subprocess" | "ffi"`
- Run parallel testing with both backends
- Performance comparison benchmarks

Phase 4: FFI Default

- Switch default to FFI
- Deprecate subprocess backend
- Remove subprocess code in future major version

Documentation Standards

- **README.md**: Overview, quick start, links to detailed docs
- **docs/API.md**: Full OpenAPI spec, examples
- **docs/DEVELOPMENT.md**: Local setup, architecture overview
- **docs/SECURITY.md**: Threat model, security practices
- **docs/ADR/**: One file per significant decision, use template

---

Versioning and Releases

- **Semantic Versioning**: MAJOR.MINOR.PATCH
- **Changelog**: Keep CHANGELOG.md updated
- **Git Tags**: Tag releases (v1.0.0, v1.1.0, etc.)
- **Release Notes**: Auto-generate from changelog
- **API Versioning**: Consider `/v1/` prefix for major API changes

---

Next Steps

1. **Initialize Workspace**: Create `Cargo.toml` workspace manifest
2. **Scaffold Crates**: Use `cargo new --lib` for each crate
3. **Define Traits**: Start with `clara-core` traits and types
4. **Implement Subprocess Backend**: `clara-clips/src/backend/subprocess/`
5. **Build API Layer**: `clara-api` with basic routes
6. **Add Configuration**: `clara-config` with validation
7. **Implement Security**: `clara-security` filters and auth
8. **Add Observability**: `clara-metrics` for logging/metrics
9. **Write Tests**: Integration suite in `tests/`
10. **Document**: Complete README and API docs

Appendix: File Naming Conventions

- **Rust files**: `snake_case.rs`
- **CLIPS files**: `snake_case.clp`
- **Config files**: `lowercase.toml`
- **Docs**: `UPPERCASE.md` for top-level, `kebab-case.md` for subdocs
- **Test files**: Match source with `_tests.rs` suffix or integration test name

Appendix: Key Dependencies (Estimated)

clara-api

- axum (HTTP server)
- tower (middleware)
- serde, serde_json
- tokio

clara-clips

- tokio (async I/O)
- regex (output parsing)
- bindgen (FFI, future)

clara-security

- jsonwebtoken (JWT)
- argon2 (password hashing, if needed)
- seccomp (sandboxing, optional)

clara-metrics

- prometheus (metrics)
- tracing, tracing-subscriber
- opentelemetry (optional)

clara-persistence

- serde (serialization)
- flate2 (compression)
- aes-gcm (encryption)

This layout provides a solid foundation for Clara's Cerebrum that is modular, testable, secure, and ready to evolve from subprocess to FFI integration while maintaining clean architectural boundaries.