Title: Developing an OpenSSH-Compatible Command-Line Interface for libssh

Proposal Summary:

While libssh provides a powerful library for SSH client and server communication, it currently lacks a general-purpose command-line interface comparable to OpenSSH's ssh binary. This project aims to build an OpenSSH-compatible CLI that leverages libssh internally and replicates the core functionality of OpenSSH's CLI, such as user authentication, command execution, and file transfer. This tool would serve as both a proof of completeness for libssh and a functional alternative to OpenSSH for users and developers. Deliverables will include a robust C-based CLI tool, test coverage for core SSH functions, compatibility documentation, and feedback-driven feature parity analysis.

Project Size: Large (350 hours)

Project Technologies: C, libssh, POSIX Sockets, OpenSSH CLI behavior, SSH protocol, Valgrind, Make, Git

Project Topics: Cybersecurity, Networking, CLI Tools, Protocol Implementation, Linux Tools

Technology Stack and Approach:

Tools and Languages:

- Language: C (ISO C99 or newer)
- Core Library: libssh (client mode)
- Build Tools: CMake, Make
- Testing Tools: manual testing scripts, possibly unit test integration using cmocka
- Debugging: Valgrind, GDB
- Reference Material: OpenSSH man pages, particularly ssh(1)

Development Approach:

1. Research and Comparison: Analyze key use cases and flags supported by the ssh command. Document which features libssh supports and which require new implementation.

2. CLI Interface Implementation: Build a new command-line front-end using argp or getopt_long() to parse OpenSSH-compatible flags.

3. Fallback & Graceful Degradation: Where libssh lacks features, fail gracefully and log for future support.

4. Validation: Compare behavior with OpenSSH using scripted SSH calls and remote execution checks.

5. Community Feedback: Use GitHub issues to track unsupported features and plan implementation.

Deliverables:

- A standalone CLI binary using libssh (e.g., lssh)

- Support for interactive shell sessions, command execution, key authentication, port specification, and verbosity flags

- A feature coverage map comparing to OpenSSH CLI

- Unit/integration testing scripts for core functions

- Documentation including usage examples and internal structure

- Issue reports for all missing libssh features

Timeline:

Community Bonding:

- Join libssh community channels

- Study OpenSSH man pages and libssh example code

- Create feature map: OpenSSH flags vs. libssh coverage

Week 1-3:

- Build CLI skeleton using argp or getopt_long

- Implement core connection logic

- Manual test SSH login and command execution

Week 4-6:

- Expand CLI to support flags and remote commands

- Begin documenting unsupported flags and libssh gaps

- Begin automated test development

Midterm Evaluation:

- Working CLI with basic options and remote command execution

- Initial test cases and README

Week 7-9:

- Add key authentication, logging verbosity, host confirmation

- Improve error handling

Week 10-12:

- Complete documentation and test reports

- Compatibility testing with OpenSSH

- Optional: server-side parity exploration

Final Week:

- Final polish, code cleanup, documentation

- Submit public demo and optional PR

What I Bring to the Table:

I have a strong background in networking and an active interest in secure communication protocols.

I've spent significant time working with SSH as a user and network admin, including tunneling, authentication, and remote scripting. While I'm still early in my C journey and new to open source, I see this project as an ideal environment to grow by connecting systems-level knowledge with code that makes real impact.