ripgrep crates/grep/src/lib.rs: Code Companion

Reference code for the The Umbrella Crate lecture. Sections correspond to the lecture document.

Section 1: What This File Does

```
// The entire lib.rs file - remarkably minimal for such an important role
/*!
ripgrep, as a library.
...
*/

pub extern crate grep_cli as cli;
pub extern crate grep_matcher as matcher;
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;
pub extern crate grep_printer as printer;
pub extern crate grep_regex as regex;
pub extern crate grep_searcher as searcher;
```

The entire file is under 20 lines of actual code. Its sole purpose is organizational—providing a unified namespace for six specialized crates.

Section 2: The Umbrella Crate Pattern

```
// Without the umbrella pattern, users would need:
use grep_cli;
use grep_matcher;
use grep_rinter;
use grep_regex;
use grep_searcher;

// With the umbrella pattern, users get a clean hierarchy:
use grep::cli;
use grep::matcher;
use grep::printer;
use grep::regex;
use grep::searcher;
```

The as clause in each re-export strips the <code>grep_</code> prefix, transforming <code>grep_matcher</code> into simply matcher within the <code>grep</code> namespace.

Section 3: Module Documentation and the Honesty of Incomplete APIs

```
/*!
ripgrep, as a library.

This library is intended to provide a high level facade to the crates that
make up ripgrep's core searching routines. However, there is no high level
documentation available yet guiding users on how to fit all of the pieces
together.

Every public API item in the constituent crates is documented, but examples
are sparse.

A cookbook and a guide are planned.
*/
```

The [/*! ... */] syntax creates an inner doc comment—documentation that describes the containing module rather than the following item. This text appears at the top of the crate's generated documentation.

Section 4: The Six Pillars of ripgrep Search

```
// CLI utilities for search tool development
pub extern crate grep_cli as cli;

// The core Matcher trait - the abstraction all pattern engines implement
pub extern crate grep_matcher as matcher;
```

These two crates handle environment integration (cli) and define the central abstraction (matcher) that enables ripgrep's pluggable regex engine architecture.

Section 5: Regex Engines: Default and Optional

```
// Always included: Rust's native regex engine
pub extern crate grep_regex as regex;

// Conditionally compiled: only present if pcre2 feature is enabled
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;
```

The #[cfg(feature = "pcre2")] attribute means this entire line is removed from compilation when the feature is disabled. The resulting binary has no PCRE2 code at all—not disabled code, absent code.

Section 6: The Searcher and Printer: Orchestrating Results

```
// Coordinates reading content and applying matchers
pub extern crate grep_searcher as searcher;

// Handles output formatting (standard, JSON, colorized)
pub extern crate grep_printer as printer;
```

The searcher handles the "how to search" (line iteration, context tracking, memory mapping), while the printer handles "how to display" (formatting, colorization, structured output).

Section 7: The pub extern crate Pattern

In Rust 2018+, extern crate is rarely needed except for this re-export pattern. The pub visibility combined with as renaming creates a stable public interface independent of internal crate naming.

Quick Reference

Crate Mapping

	Public API	
grep_cli	grep::cli	
grep_matcher	grep::matcher	Core Matcher trait definition
grep_regex	grep::regex	
grep_pcre2	grep::pcre2	
grep_searcher	grep::searcher	
grep_printer	grep::printer	

Feature Flags

```
# In Cargo.toml of a crate depending on grep:
[dependencies]
grep = { version = "x.y", features = ["pcre2"] } # Enable PCRE2 support
```

Search Pipeline Architecture

