Makefile Cheat Sheet

Definitions:

*   **Makefile:** A file containing rules for the `make` utility to automate build processes.
*   **Target:** The file to be built (e.g., executable, object file).
*   **Prerequisites (Dependencies):** Files required to build a target.
*   **Command:** The shell command executed to build the target from its prerequisites.
*   **Macros:** Variables within a Makefile for reusability (e.g., `CC`, `CFLAGS`).

Key Formulas (Makefile Structure):

```makefile
target: prerequisite1 prerequisite2 ...
    <TAB> command
```

*   If any prerequisite is newer than the target, the command is executed.
*   `make` recompiles only necessary files based on dependencies and timestamps.

Common Macros:

*   `CC`: C Compiler (e.g., `gcc`)
*   `CFLAGS`: Compiler Flags (e.g., `-Wall -O2 -g`)
*   `LDFLAGS`: Linker Flags (e.g., `-lm`)
*   `LIBS`: Libraries to link against (e.g. `-lmath`)
*   `SRC`: Source Directory
*   `INC`: Include Directory
*   `BIN`: Binary Directory

Tips and Practices:

*   **Clean Target:**
    ```makefile
    clean:
        rm -f *.o executable
    ```

```

*   **Macros:** Use macros to avoid repetition and improve maintainability.
*   **Compiler Flags:** Utilize compiler flags for warnings, optimization, and debugging.
*   **File Organization:** Use standard directories (src, include) for source code and header files.
*   **Dependencies:** Explicitly declare all dependencies to ensure correct builds.
*   **Multiline commands:** Use backslashes `\` at the end of line to continue commands.
*   **Inspect predefined Macros:** Use `make -p`

**Common Pitfalls:**

*   **Tab vs. Spaces:**  Commands *must* start with a TAB character, not spaces.
*   **Lack of Toolchain Understanding:** Understand the compiler toolchain.
*   **Forgetting Dependencies:** Failing to list all dependencies.
*   **Not Using Macros:** Verbose and harder-to-maintain Makefiles.
*   **Undefined Macros:** Referencing undefined macros leads to unexpected behavior (null strings).
*   **Improper Makefile name:** Ensure the Makefile is named as "Makefile" or "makefile."

**Suggested Practice:**

1.  Basic Makefile (single C file)
2.  Add Header Dependencies
3.  Implement Macros
4.  Create a Clean Target
5.  Multiple Source Files with a Shared Header
6.  Linking Libraries
7.  Preprocessor Flags
8.  Debugging (the `-g` flag) and Dry Run (the `-n` flag)
9.  File Organization
10. Understanding existing Makefiles in open-source projects.
```