/dwarf-expert

Source: `~/.claude/skills/tob-dwarf-expert/skills/dwarf-expert/SKILL.md`

---

name: dwarf-expert description: Provides expertise for analyzing DWARF debug files and understanding the DWARF debug format/standard (v3-v5). Triggers when understanding DWARF information, interacting with DWARF files, answering DWARF-related questions, or working with code that parses DWARF data. allowed-tools:

- Read
- Bash
- Grep
- Glob
- WebSearch

Overview

This skill provides technical knowledge and expertise about the DWARF standard and how to interact with DWARF files. Tasks include answering questions about the DWARF standard, providing examples of various DWARF features, parsing and/or creating DWARF files, and writing/modifying/analyzing code that interacts with DWARF data.

When to Use This Skill

- Understanding or parsing DWARF debug information from compiled binaries
- Answering questions about the DWARF standard (v3, v4, v5)
- Writing or reviewing code that interacts with DWARF data
- Using `dwarfdump` or `readelf` to extract debug information
- Verifying DWARF data integrity with `llvm-dwarfdump --verify`
- Working with DWARF parsing libraries (libdwarf, pyelftools, gimli, etc.)

When NOT to Use This Skill

- **DWARF v1/v2 Analysis**: Expertise limited to versions 3, 4, and 5.
- **General ELF Parsing**: Use standard ELF tools if DWARF data isn't needed.
- **Executable Debugging**: Use dedicated debugging tools (gdb, lldb, etc) for debugging executable code/runtime behavior.
- **Binary Reverse Engineering**: Use dedicated RE tools (Ghidra, IDA) unless specifically analyzing DWARF sections.
- **Compiler Debugging**: DWARF generation issues are compiler-specific, not covered here.

Authoritative Sources

When specific DWARF standard information is needed, use these authoritative sources:

1. **Official DWARF Standards (dwarfstd.org)**: Use web search to find specific sections of the official DWARF specification at dwarfstd.org. Search queries like "DWARF5 DW_TAG_subprogram attributes site:dwarfstd.org" are effective.
2. **LLVM DWARF Implementation**: The LLVM project's DWARF handling code at `llvm/lib/DebugInfo/DWARF/` serves as a reliable reference implementation. Key files include:
   - `DWARFDie.cpp` - DIE handling and attribute access
   - `DWARFUnit.cpp` - Compilation unit parsing
   - `DWARFDebugLine.cpp` - Line number information
   - `DWARFVerifier.cpp` - Validation logic
3. **libdwarf**: The reference C implementation at github.com/davea42/libdwarf-code provides detailed handling of DWARF data structures.

Verification Workflows

Use `llvm-dwarfdump` verification options to validate DWARF data integrity:

Structural Validation

```
# Verify DWARF structure (compile units, DIE relationships, address ranges)
llvm-dwarfdump --verify <binary>

# Detailed error output with summary
llvm-dwarfdump --verify --error-display=full <binary>
```

```
# Machine-readable JSON error summary
llvm-dwarfdump --verify --verify-json=errors.json <binary>
```

Quality Metrics

```
# Output debug info quality metrics as JSON
llvm-dwarfdump --statistics <binary>
```

The `--statistics` output helps compare debug info quality across compiler versions and optimization levels.

Common Verification Patterns

- **After compilation**: Verify binaries have valid DWARF before distribution
- **Comparing builds**: Use `--statistics` to detect debug info quality regressions
- **Debugging debuggers**: Identify malformed DWARF causing debugger issues
- **DWARF tool development**: Validate parser output against known-good binaries

Parsing DWARF Debug Information

readelf

ELF files can be parsed via the `readelf` command ({baseDir}/reference/readelf.md). Use this for general ELF information, but prefer `dwarfdump` for DWARF-specific parsing.

dwarfdump

DWARF files can be parsed via the `dwarfdump` command, which is more effective at parsing and displaying complex DWARF information than `readelf` and should be used for most DWARF parsing tasks ({baseDir}/reference/dwarfdump.md).

Working With Code

This skill supports writing, modifying, and reviewing code that interacts with DWARF data. This may involve code that parses DWARF debug data from scratch or code that leverages libraries to parse and interact with DWARF data ({baseDir}/reference/coding.md).

Choosing Your Approach

```
┌─ Need to verify DWARF data integrity?
│   └─ Use `llvm-dwarfdump --verify` (see Verification Workflows above)
├─ Need to answer questions about the DWARF standard?
│   └─ Search dwarfstd.org or reference LLVM/libdwarf source
├─ Need simple section dump or general ELF info?
│   └─ Use `readelf` ({baseDir}/reference/readelf.md)
├─ Need to parse, search, and/or dump DWARF DIE nodes?
│   └─ Use `dwarfdump` ({baseDir}/reference/dwarfdump.md)
└─ Need to write, modify, or review code that interacts with DWARF data?
    └─ Refer to the coding reference ({baseDir}/reference/coding.md)
```

Revision #5
Created 2026-02-18 08:40:06 UTC by John
Updated 2026-07-05 20:01:46 UTC by John