Advanced GDB Cheat Sheet

Threads & Concurrency

Command	Description
`info threads`	List threads with IDs, states, top frame.
`thread <n>`	Switch to thread n.
`thread apply all bt full`	Backtrace for all threads with locals (deadlock detection).
`info thread <n>`	Detailed info about thread n.
`info threads -full`	More verbose thread info (needs libthread_db).
`set scheduler-locking step`	Step only current thread (freeze others).
`info tasks`	Show Linux tasks (kernel threads) if supported.
`monitor info threads`	Extra info with gdbserver/remote.
`info pthread` *(glibc)*	Show pthread structures (if available).

Deadlock check:

1. `info threads` – see which threads are waiting.
2. `thread apply all bt full` – look for `pthread_mutex_lock`, `futex_wait`.
3. Correlate owner TIDs from mutex structs to thread TIDs.

Processes & Fork/Exec

Command	Description
`set follow-fork-mode parent|child`	Debug parent (default) or child after fork.
`set detach-on-fork on|off`	Keep parent/child under control after fork.
`info inferiors`	List processes (inferiors) under GDB.
`inferior <n>`	Switch between inferiors.
`add-inferior`	Add empty inferior slot (multi-process debugging).
`clone-inferior <n>`	Duplicate state of inferior n.
`detach inferior <n>`	Detach process n.

Watchpoints & Advanced Breakpoints

Command	Description
`watch <expr>`	Break when expr changes.
`rwatch <expr>`	Break when expr is read.
`awatch <expr>`	Break on read or write.
`catch fork`	Break on `fork()`.
`catch exec`	Break on `exec()`.
`catch syscall <n>`	Break on syscall number n.
`catch throw` / `catch catch`	Break on C++ exception throw/catch.
`break if <cond>`	Conditional breakpoint.
`commands <n>`	Define actions when breakpoint n triggers.

Signals

Command	Description
`handle <sig> print|ignore|stop`	Control GDB's reaction to signals.
`info signals`	List signals and handling status.
`signal <sig>`	Send signal to program.
`catch signal`	Break when program receives a signal.

Expressions & Data

Command	Description
`print /x <expr>`	Print in hex.
`ptype <expr>`	Show type.
`whatis <expr>`	Shorthand type query.
`set var <expr>=<val>`	Assign variable.
`call <func>(args)`	Call function manually.
`print *(type*)addr`	Cast and dereference memory.
`x/<nfu> <addr>`	Examine memory (n=count, f=format, u=unit).
`set print pretty on`	Pretty-print arrays/structs.
`set print elements 0`	Don't truncate long arrays.

Advanced Navigation

Command	Description
`tui enable`	Enable TUI (text UI) mode.
`layout src|asm|regs`	Change TUI window.
`focus src|asm|regs`	Move focus in TUI.
`disassemble /m <func>`	Mixed source/assembly view.
`record` / `record full`	Start process recording.
`reverse-continue`	Run backward in record mode.
`reverse-step` / `reverse-next`	Step backward.

Remote Debugging & Core Files

Command	Description
`target remote <host:port>`	Connect to gdbserver.
`target extended-remote <...>`	Extended remote mode.
`monitor <cmd>`	Send raw command to gdbserver.
`set sysroot <path>`	Point to target system libraries.
`core <corefile>`	Load a core dump.
`info files`	Show loaded executable and symbols.

Scripting & Automation

Command	Description
`source <file>`	Run GDB script.
`define <cmd>`	Define a custom command.
`document <cmd>`	Add help text to your custom command.
`python ... end`	Inline Python scripting.
`set logging on`	Log session to gdb.txt.
`macro expand <expr>`	Expand C macros.
`macro define`	Define C macros inside GDB.

Mutex Ownership & Deadlock Helpers

Command / Tool	Description
`print *(pthread_mutex_t *)<ptr>`	Inspect mutex struct; look at `__data.__owner` for owning TID (glibc).
`info threads`	Match owner TID to GDB thread number.
`thread apply all bt full`	Full stacks to find blockers/waiters.

gdb-pthread-utils (Python extension)

github.com/gbpthreads/gdb-pthread-utils

Command (after loading plugin)	Description
`pthread info mutex <ptr>`	Pretty-print mutex state and owner thread.
`pthread info all-mutexes`	Show all known mutexes and their owners.
`pthread info cond <ptr>`	Show condition variable waiters.

Load plugin:

```
source /path/to/gdb-pthread-utils.py
```

Popular GDB Python Extensions

URL / Source	What It Adds
github.com/longld/peda	Enhanced disassembly, registers, shellcode tools.
github.com/pwndbg/pwndbg	Modern replacement for PEDA; better heap/threads view.
github.com/hugsy/gef	Modular enhancement for reverse-engineering/exploit dev.
github.com/snare/voltron	External TUI panels in your terminal.
github.com/cloudburst/libheap	Heap debugging for glibc (malloc/free).
github.com/gbpthreads/gdb-pthread-utils	Pretty-print pthread mutexes/conds, show owners/waiters.
github.com/cyrus-and/gdb-dashboard	Dashboard with registers, stack, source, breakpoints.

rr-project.org	Record/replay debugging with reverse exec execution.