Debugging

2022.04.25

SWPP Practice Session

Seunghyeon Nam (with lots of derived works)

Typical Bugfix Process

- Notice an error
- Narrow down the line that causes the error
    - If the program crashes, look for the assertion or invalid pointer
    - If the program yields wrong output, look for the output variable
- Traceback to the line that started to go wrong

Narrowing Down the Line

- How do you locate the exact line that crashes?
  - Guess the location
  - Insert some `std::cout` or `std::cerr` all over the code
  - Rebuild
  - Look for the last printed message
  - Repeat until you actually pinpoint the location

Narrowing Down the Line

- This is horribly inefficient!
  - Taking a wild guess in a large codebase purely depends on luck
  - Rebuilding a large codebase may take minutes or even hours
  - And you have to repeat it until you actually find the bug
- Locating a single bug may already take hours or days

Traceback

- Most of the errors cannot be fixed locally
    - It is likely that the code that ‘causes’ the error is not a bug
    - The code that ‘leads to’ the error is the real verdict
    - But these two are usually far away from each other…
- You have to locate the code that first went wrong
    - Narrow down, take a step back, narrow down, again and again

Debugger to the Rescue

- Debugger can control the execution of your program
  - Line by line
  - In and out of function
  - Pause on assertion, throw, catch, breakpoint

Debugger to the Rescue

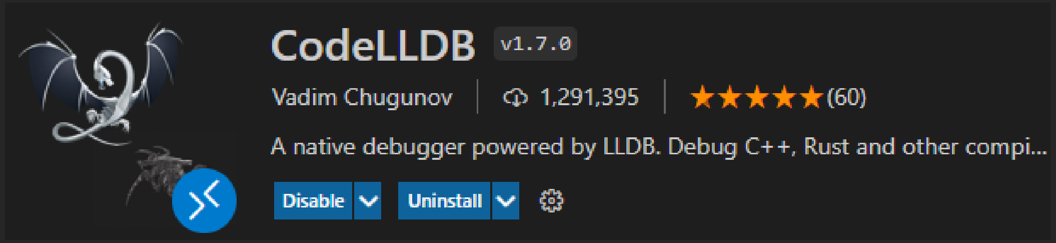
- Debugger can expose the execution context of your program
    - Call stack
    - Local/global variables and values

Using the Debugger

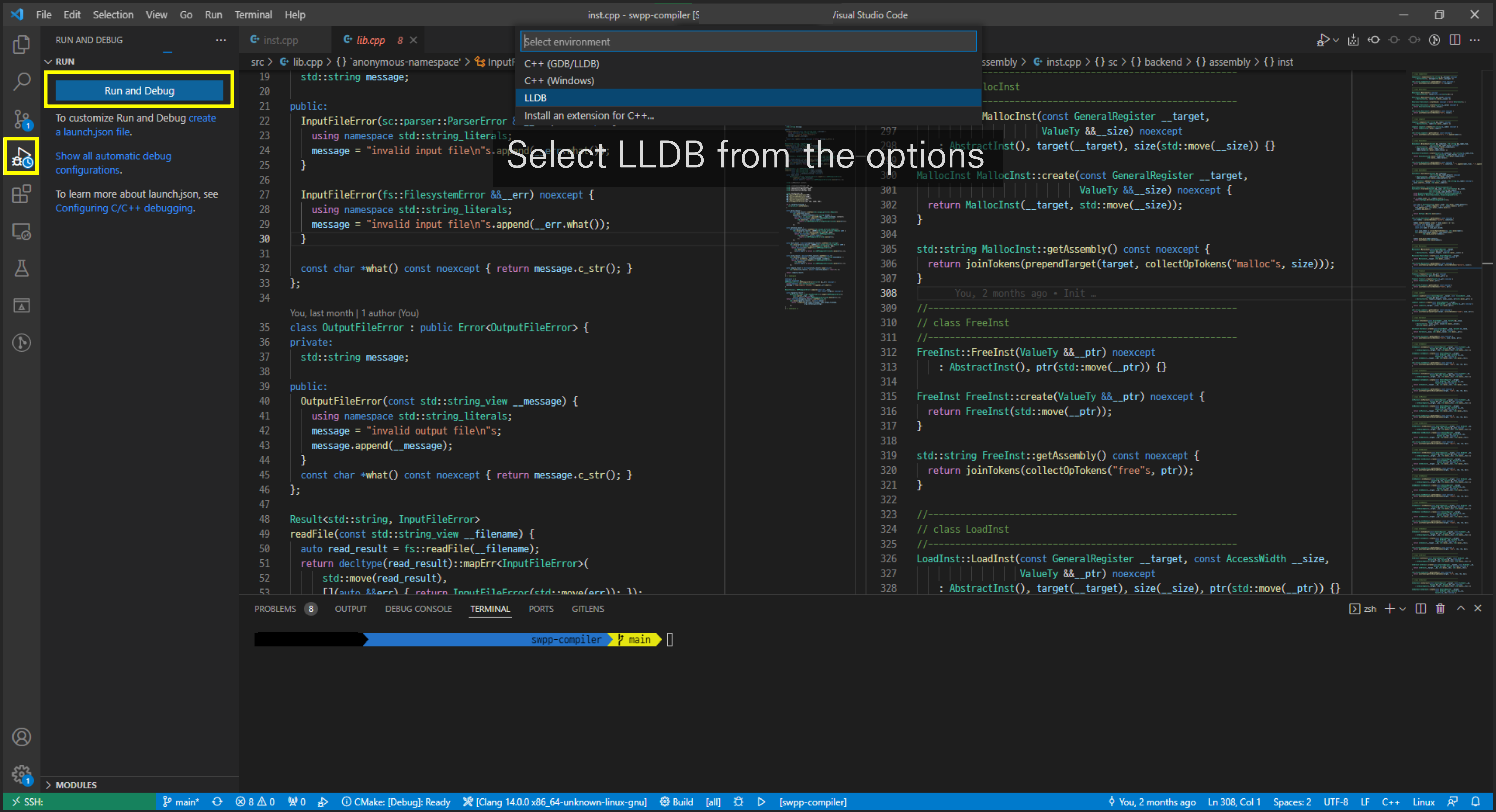
- LLDB: LLVM debugger
  - You have to enable LLDB project when building the LLVM
  - Already included if you built the LLVM using class repo script
- You must build your program with clang
- You must build your program in debug mode

Using the Debugger

- We'll use vscode extension for convenience
  - CodeLLDB


CodeLLDB v1.7.0
Vadim Chugunov | 1,291,395 | ★★★★★(60)
A native debugger powered by LLDB. Debug C++, Rust and other compi...

- LLDB directory should be added to your PATH
  - What is PATH?



Select LLDB from the options

File Edit Selection View Go Run Terminal Help — inst.cpp - swpp-compiler [S /isual Studio Code

RUN AND DEBUG

RUN

Run and Debug

To customize Run and Debug create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

Select environment

- C++ (GDB/LLDB)
- C++ (Windows)
- LLDB
- Install an extension for C++...

src > lib.cpp > {} `anonymous-namespace' > InputF

```cpp
  std::string message;

public:
  InputFileError(sc::parser::ParserError &
    using namespace std::string_literals;
    message = "invalid input file\n"s.ap
  }

  InputFileError(fs::FilesystemError &&__err) noexcept {
    using namespace std::string_literals;
    message = "invalid input file\n"s.append(__err.what());
  }

  const char *what() const noexcept { return message.c_str(); }
};

You, last month | 1 author (You)
class OutputFileError : public Error<OutputFileError> {
private:
  std::string message;

public:
  OutputFileError(const std::string_view __message) {
    using namespace std::string_literals;
    message = "invalid output file\n"s;
    message.append(__message);
  }
  const char *what() const noexcept { return message.c_str(); }
};

Result<std::string, InputFileError>
readFile(const std::string_view __filename) {
  auto read_result = fs::readFile(__filename);
  return decltype(read_result)::mapErr<InputFileError>(
      std::move(read_result),
```

ssembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst

```cpp
locInst
//------------------------------------------------------------
MallocInst(const GeneralRegister __target,
                     ValueTy &&__size) noexcept
    : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}
      You, 2 months ago • Init …
//------------------------------------------------------------
// class FreeInst
//------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//------------------------------------------------------------
// class LoadInst
//------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
    : AbstractInst(), target(__target), size(__size), ptr(std::move(__ptr)) {}
```

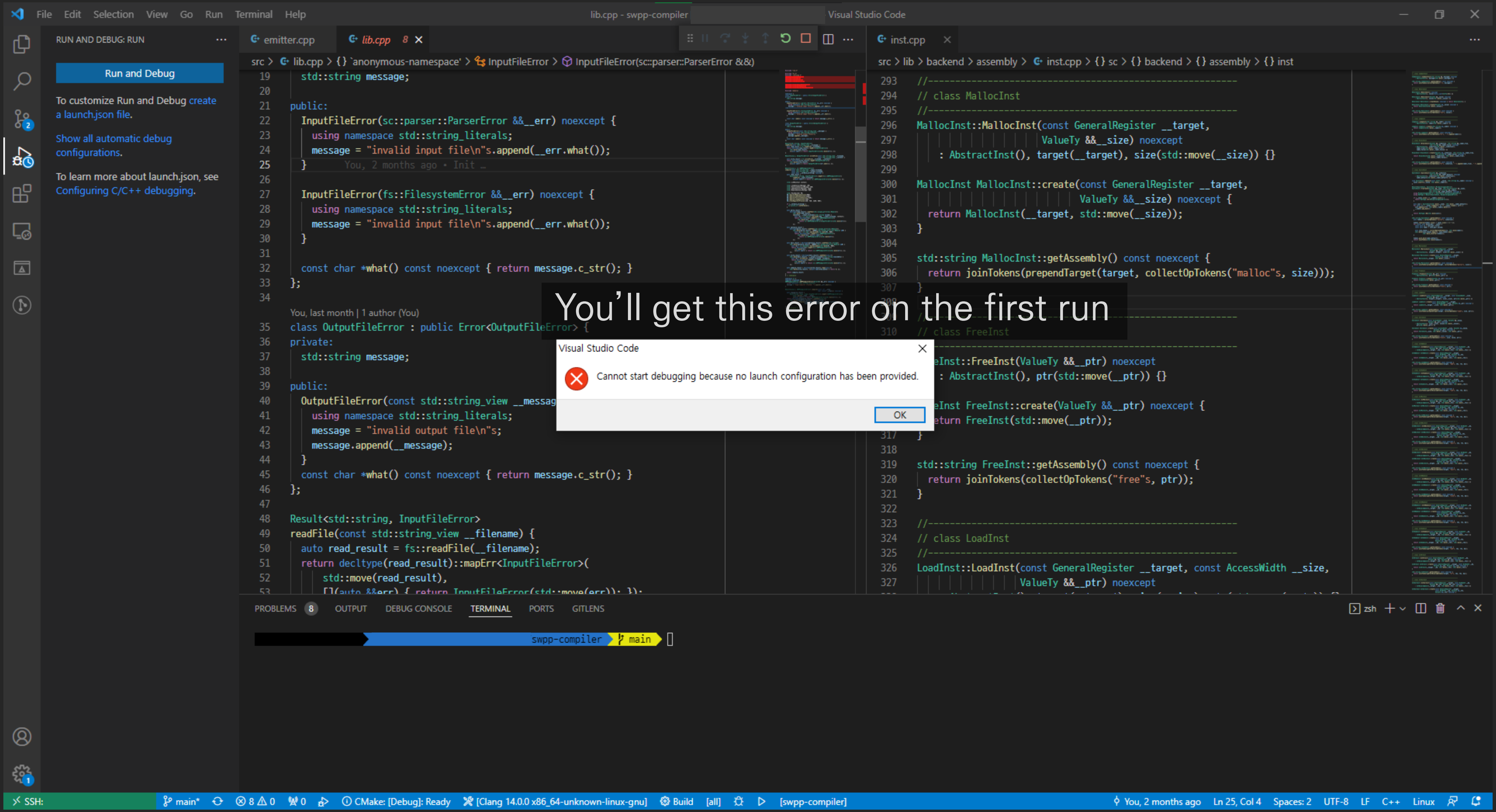
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[redacted] swpp-compiler  main
```

MODULES

SSH: | main* | 8 ⚠ 0 | 0 | CMake: [Debug]: Ready | [Clang 14.0.0 x86_64-unknown-linux-gnu] | Build | [all] | [swpp-compiler] | You, 2 months ago | Ln 308, Col 1 | Spaces: 2 | UTF-8 | LF | C++ | Linux

You’ll get this error on the first run



Visual Studio Code

Cannot start debugging because no launch configuration has been provided.

OK

RUN AND DEBUG: RUN

Run and Debug

To customize Run and Debug create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

src > lib.cpp > {} `anonymous-namespace' > InputFileError > InputFileError(sc::parser::ParserError &&)

```cpp
  std::string message;

public:
  InputFileError(sc::parser::ParserError &&__err) noexcept {
    using namespace std::string_literals;
    message = "invalid input file\n"s.append(__err.what());
  }       You, 2 months ago • Init …

  InputFileError(fs::FilesystemError &&__err) noexcept {
    using namespace std::string_literals;
    message = "invalid input file\n"s.append(__err.what());
  }

  const char *what() const noexcept { return message.c_str(); }
};

You, last month | 1 author (You)
class OutputFileError : public Error<OutputFileError> {
private:
  std::string message;

public:
  OutputFileError(const std::string_view __messag
    using namespace std::string_literals;
    message = "invalid output file\n"s;
    message.append(__message);
  }
  const char *what() const noexcept { return message.c_str(); }
};

Result<std::string, InputFileError>
readFile(const std::string_view __filename) {
  auto read_result = fs::readFile(__filename);
  return decltype(read_result)::mapErr<InputFileError>(
      std::move(read_result),
```

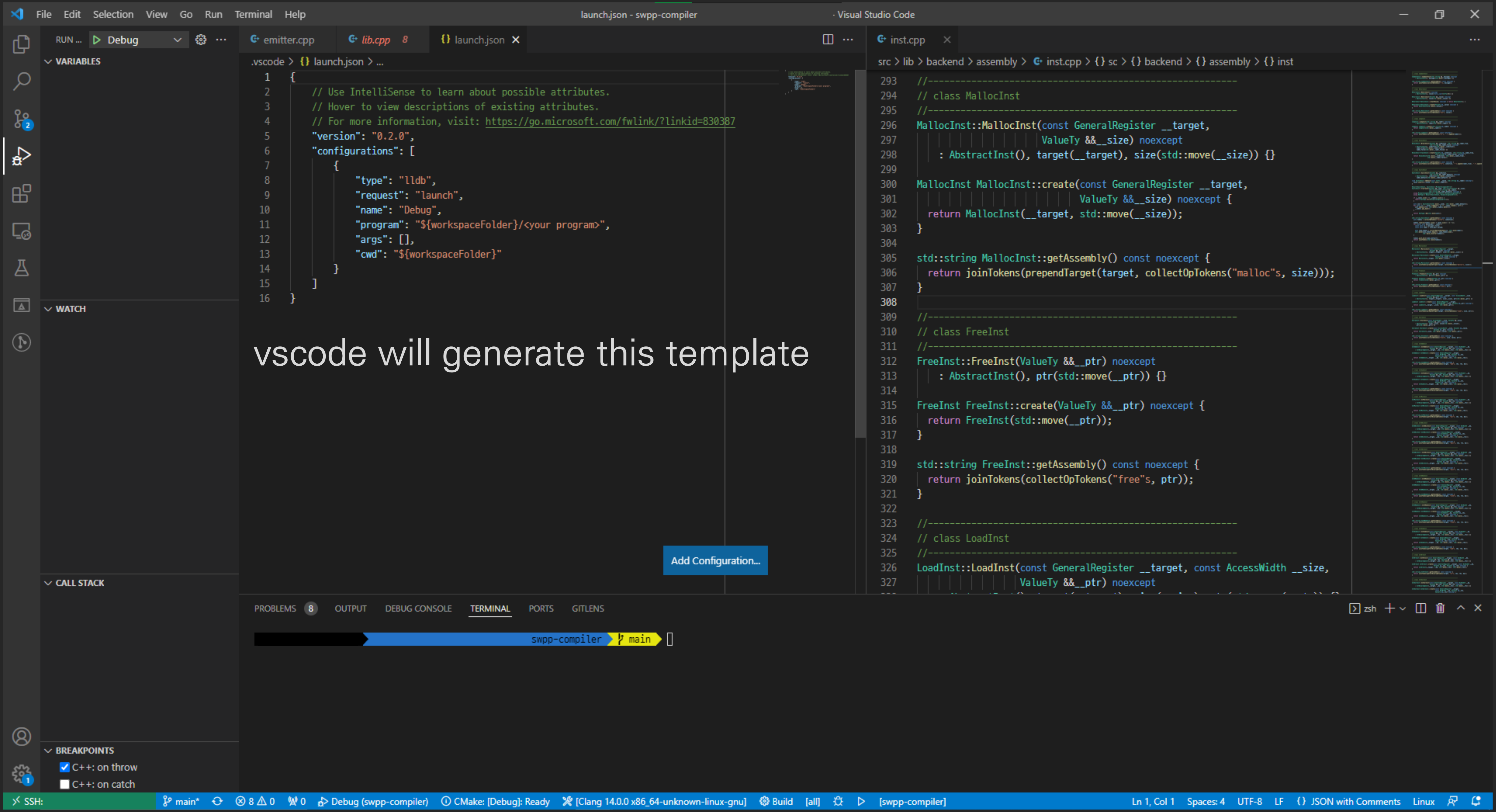
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst

```cpp
//-----------------------------------------------------------
// class MallocInst
//-----------------------------------------------------------
MallocInst::MallocInst(const GeneralRegister __target,
                       ValueTy &&__size) noexcept
    : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//-----------------------------------------------------------
// class FreeInst
//-----------------------------------------------------------
eInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

eInst FreeInst::create(ValueTy &&__ptr) noexcept {
eturn FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//-----------------------------------------------------------
// class LoadInst
//-----------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
swpp-compiler  main
```

SSH: | main* | ⊗ 8 ⚠ 0 | 0 | CMake: [Debug]: Ready | [Clang 14.0.0 x86_64-unknown-linux-gnu] | Build | [all] | [swpp-compiler] | You, 2 months ago | Ln 25, Col 4 | Spaces: 2 | UTF-8 | LF | C++ | Linux

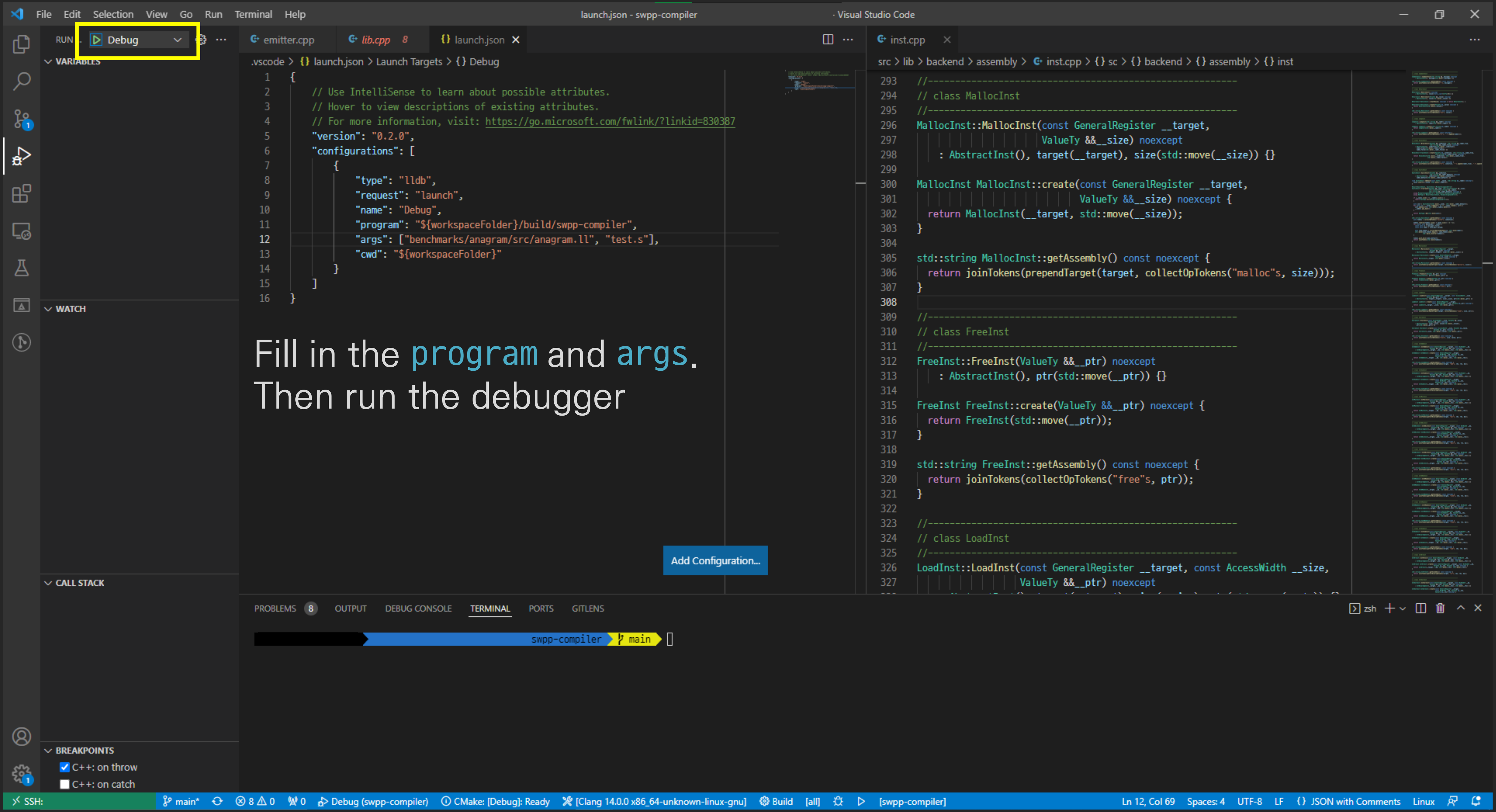
File Edit Selection View Go Run Terminal Help
lib.cpp - swpp-compiler - Visual Studio Code
emitter.cpp
lib.cpp 8
inst.cpp
PROBLEMS 8
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
zsh



vscode will generate this template

```json
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "lldb",
            "request": "launch",
            "name": "Debug",
            "program": "${workspaceFolder}/<your program>",
            "args": [],
            "cwd": "${workspaceFolder}"
        }
    ]
}
```

```cpp
//------------------------------------------------------------
// class MallocInst
//------------------------------------------------------------
MallocInst::MallocInst(const GeneralRegister __target,
                       ValueTy &&__size) noexcept
    : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//------------------------------------------------------------
// class FreeInst
//------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//------------------------------------------------------------
// class LoadInst
//------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```



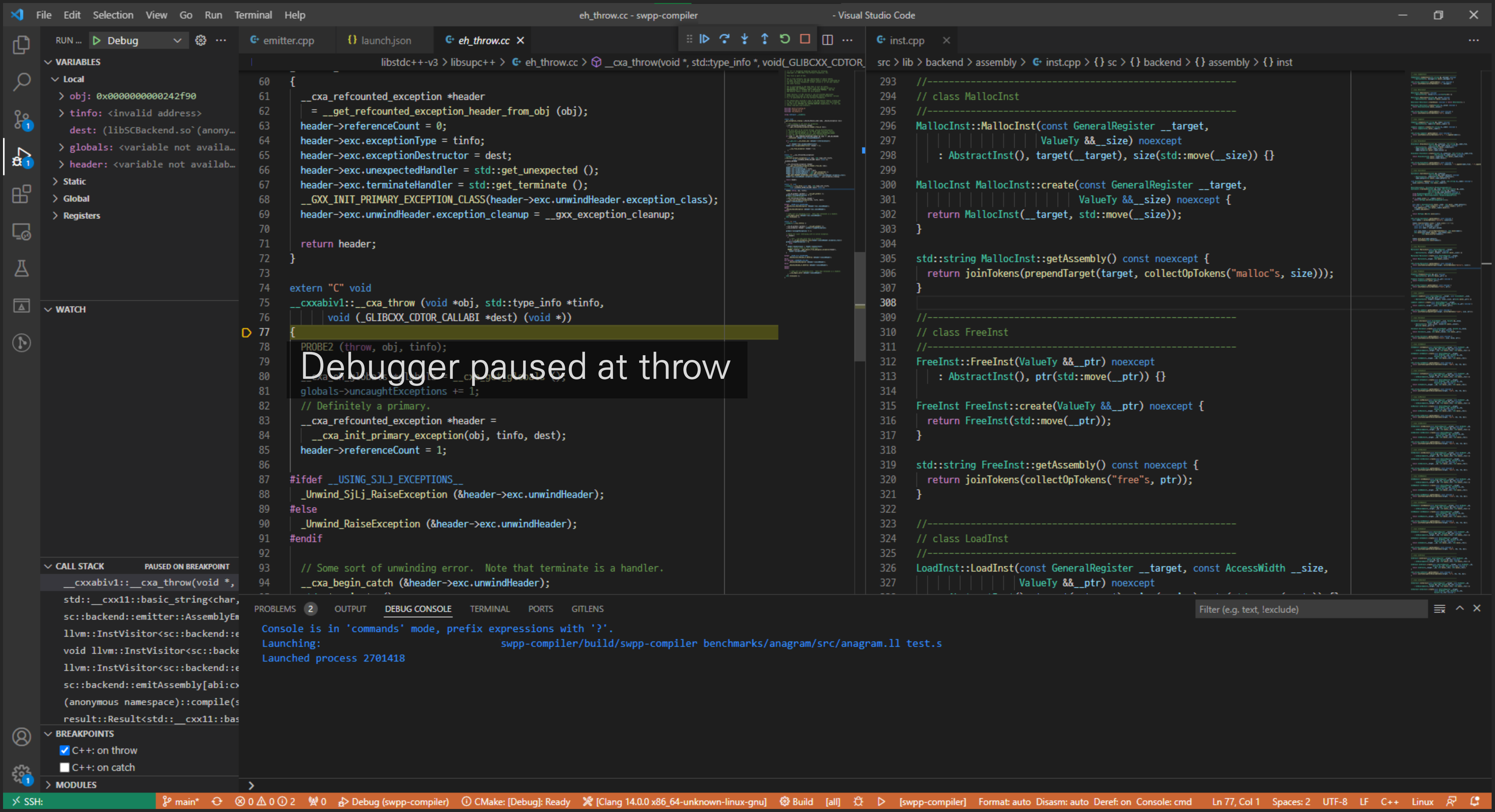
```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "type": "lldb",
      "request": "launch",
      "name": "Debug",
      "program": "${workspaceFolder}/build/swpp-compiler",
      "args": ["benchmarks/anagram/src/anagram.ll", "test.s"],
      "cwd": "${workspaceFolder}"
    }
  ]
}
```

Fill in the program and args.
Then run the debugger

```
//-------------------------------------------------------------
// class MallocInst
//-------------------------------------------------------------
MallocInst::MallocInst(const GeneralRegister __target,
                       ValueTy &&__size) noexcept
  : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//-------------------------------------------------------------
// class FreeInst
//-------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
  : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//-------------------------------------------------------------
// class LoadInst
//-------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
swpp-compiler  main
```

File Edit Selection View Go Run Terminal Help
launch.json - swpp-compiler - Visual Studio Code
RUN Debug
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
C++: on throw
C++: on catch
emitter.cpp
lib.cpp 8
launch.json
inst.cpp
.vscode > launch.json > Launch Targets > Debug
src > lib > backend > assembly > inst.cpp > sc > backend > assembly > inst
Add Configuration...
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
zsh
SSH: main* 8 0 0 Debug (swpp-compiler) CMake: [Debug]: Ready [Clang 14.0.0 x86_64-unknown-linux-gnu] Build [all] [swpp-compiler] Ln 12, Col 69 Spaces: 4 UTF-8 LF JSON with Comments Linux



Debugger paused at throw

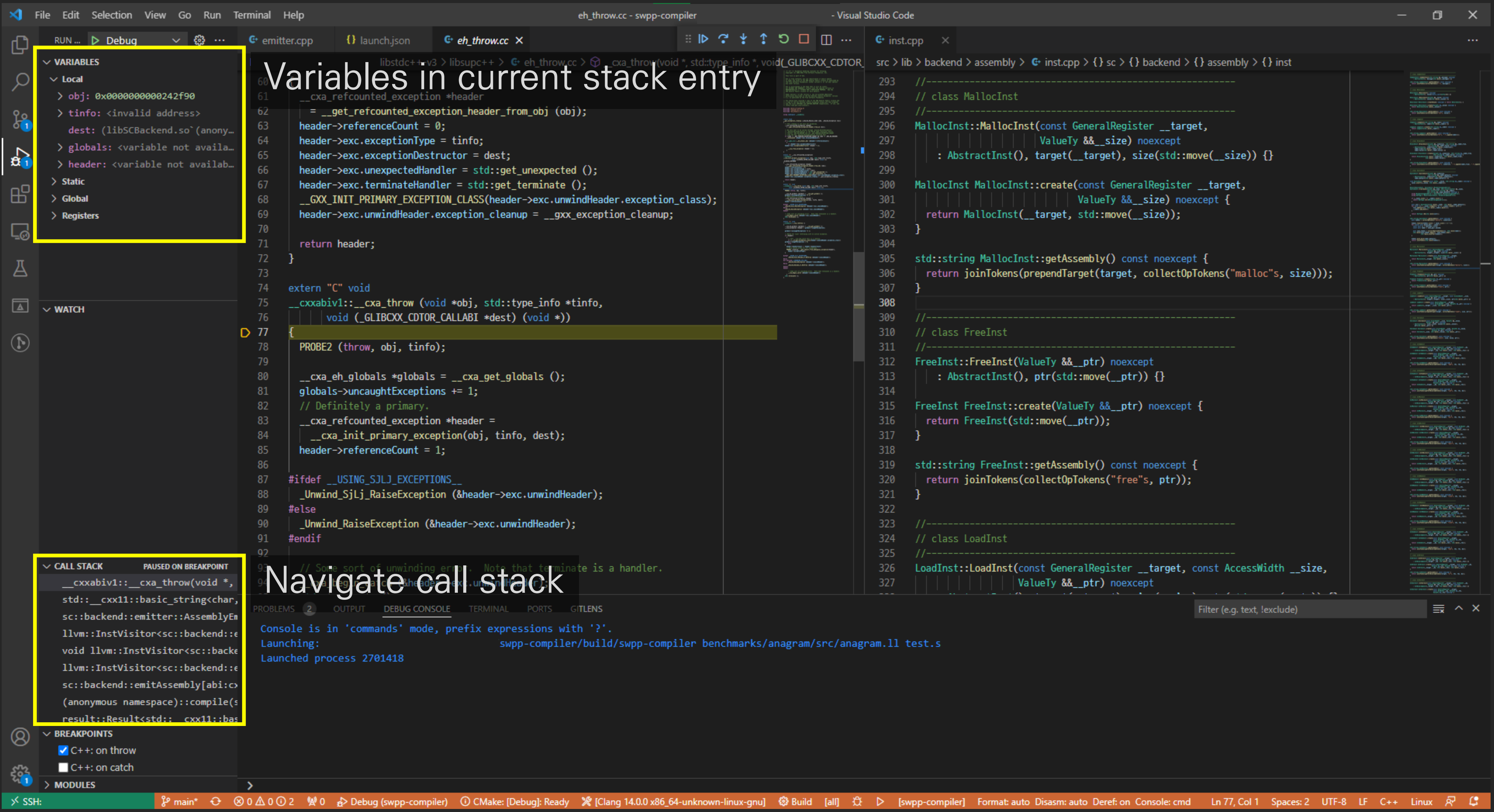
```
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_from_obj (obj);
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.unwindHeader.exception_class);
  header->exc.unwindHeader.exception_cleanup = __gxx_exception_cleanup;

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
                         void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  [illegible]
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

```
//------------------------------------------------------------
// class MallocInst
//------------------------------------------------------------
MallocInst::MallocInst(const GeneralRegister __target,
                       ValueTy &&__size) noexcept
    : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//------------------------------------------------------------
// class FreeInst
//------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//------------------------------------------------------------
// class LoadInst
//------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
VARIABLES
  Local
    obj: 0x0000000000242f90
    tinfo: <invalid address>
    dest: (libSCBackend.so`(anony...
    globals: <variable not availa...
    header: <variable not availab...
  Static
  Global
  Registers

CALL STACK                    PAUSED ON BREAKPOINT
  __cxxabiv1::__cxa_throw(void *,
  std::__cxx11::basic_string<char,
  sc::backend::emitter::AssemblyEm
  llvm::InstVisitor<sc::backend::e
  void llvm::InstVisitor<sc::backe
  llvm::InstVisitor<sc::backend::e
  sc::backend::emitAssembly[abi:cx
  (anonymous namespace)::compile(s
  result::Result<std::__cxx11::bas

BREAKPOINTS
  [✓] C++: on throw
  [ ] C++: on catch
```

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                    swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

eh_throw.cc - swpp-compiler - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN ... Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
WATCH
MODULES
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude)
SSH:
main*
Debug (swpp-compiler)
CMake: [Debug]: Ready
[Clang 14.0.0 x86_64-unknown-linux-gnu]
Build [all]
[swpp-compiler]
Format: auto Disasm: auto Deref: on Console: cmd
Ln 77, Col 1 Spaces: 2 UTF-8 LF C++ Linux


File Edit Selection View Go Run Terminal Help
eh_throw.cc - swpp-compiler - Visual Studio Code
RUN ... Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
WATCH
BREAKPOINTS
MODULES
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Filter (e.g. text, !exclude)

Variables in current stack entry

Navigate call stack

```
VARIABLES
  Local
    > obj: 0x0000000000242f90
    > tinfo: <invalid address>
      dest: (libSCBackend.so`(anony…
    > globals: <variable not availa…
    > header: <variable not availab…
  > Static
  > Global
  > Registers
```

```
CALL STACK                PAUSED ON BREAKPOINT
  __cxxabiv1::__cxa_throw(void *,
  std::__cxx11::basic_string<char,
  sc::backend::emitter::AssemblyEm
  llvm::InstVisitor<sc::backend::e
  void llvm::InstVisitor<sc::backe
  llvm::InstVisitor<sc::backend::e
  sc::backend::emitAssembly[abi:cx
  (anonymous namespace)::compile(s
  result::Result<std::__cxx11::bas
```

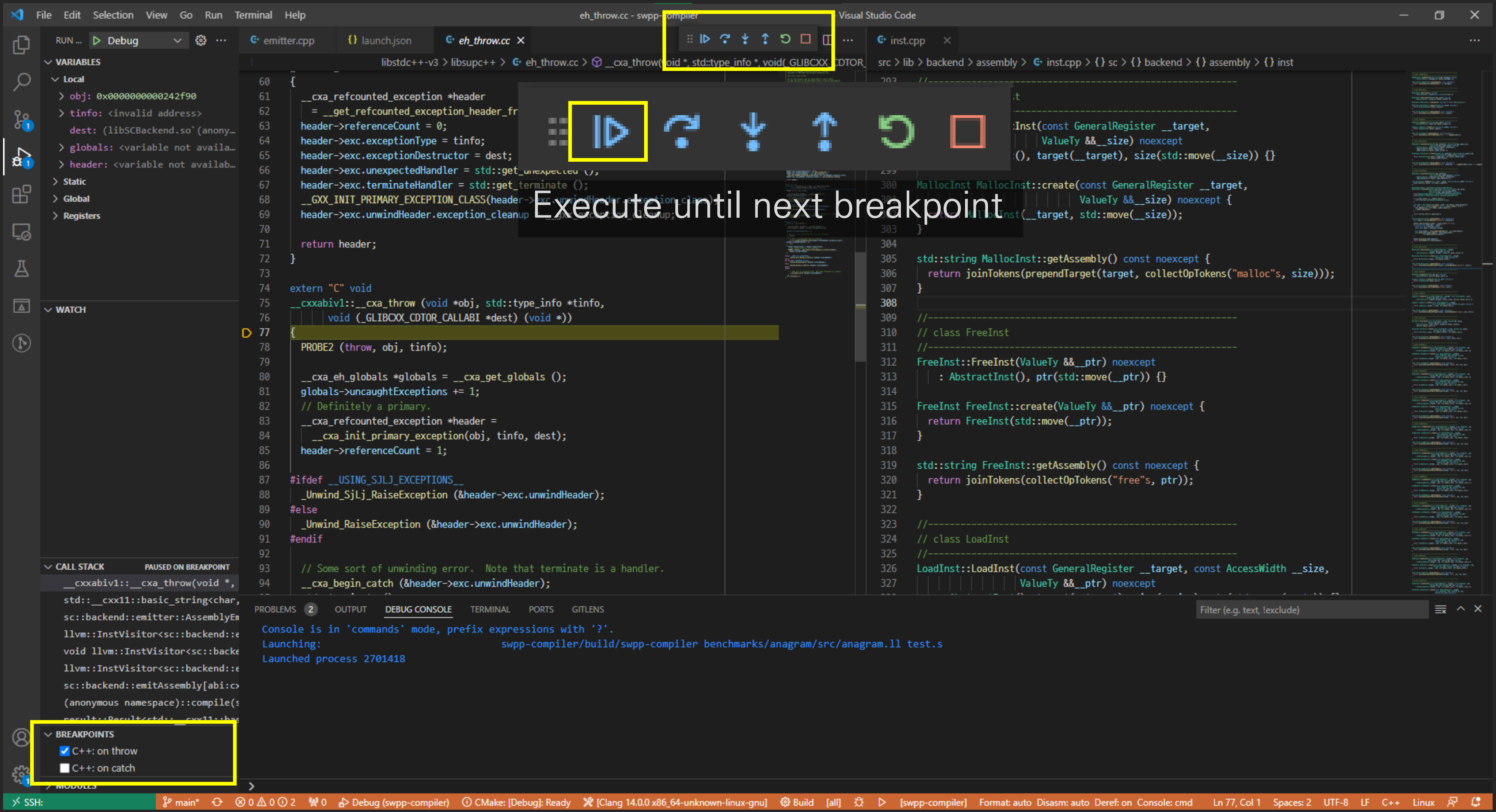
☑ C++: on throw
☐ C++: on catch

```
    __cxa_refcounted_exception *header
      = __get_refcounted_exception_header_from_obj (obj);
    header->referenceCount = 0;
    header->exc.exceptionType = tinfo;
    header->exc.exceptionDestructor = dest;
    header->exc.unexpectedHandler = std::get_unexpected ();
    header->exc.terminateHandler = std::get_terminate ();
    __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.unwindHeader.exception_class);
    header->exc.unwindHeader.exception_cleanup = __gxx_exception_cleanup;

    return header;
  }

  extern "C" void
  __cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
                           void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
  {
    PROBE2 (throw, obj, tinfo);

    __cxa_eh_globals *globals = __cxa_get_globals ();
    globals->uncaughtExceptions += 1;
    // Definitely a primary.
    __cxa_refcounted_exception *header =
      __cxa_init_primary_exception(obj, tinfo, dest);
    header->referenceCount = 1;

  #ifdef __USING_SJLJ_EXCEPTIONS__
    _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
  #else
    _Unwind_RaiseException (&header->exc.unwindHeader);
  #endif

    // Some sort of unwinding error.  Note that terminate is a handler.
```

```
//------------------------------------------------------------
// class MallocInst
//------------------------------------------------------------
MallocInst::MallocInst(const GeneralRegister __target,
                       ValueTy &&__size) noexcept
    : AbstractInst(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//------------------------------------------------------------
// class FreeInst
//------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//------------------------------------------------------------
// class LoadInst
//------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                    swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

SSH: | main* | ⊗ 0 ⚠ 0 ⓘ 2 | 0 | Debug (swpp-compiler) | CMake: [Debug]: Ready | [Clang 14.0.0 x86_64-unknown-linux-gnu] | Build | [all] | [swpp-compiler] | Format: auto Disasm: auto Deref: on Console: cmd | Ln 77, Col 1 | Spaces: 2 | UTF-8 | LF | C++ | Linux


File Edit Selection View Go Run Terminal Help
eh_throw.cc - swpp-compiler - Visual Studio Code
RUN ... Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
VARIABLES
WATCH
CALL STACK
PAUSED ON BREAKPOINT
BREAKPOINTS
MODULES
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude)

Execute until next breakpoint

```
Local
  > obj: 0x0000000000242f90
  > tinfo: <invalid address>
    dest: (libSCBackend.so`(anony…
  > globals: <variable not availa…
  > header: <variable not availab…
> Static
> Global
> Registers
```

```
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_fr
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->
  header->exc.unwindHeader.exception_cleanup

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
      void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  __cxa_eh_globals *globals = __cxa_get_globals ();
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

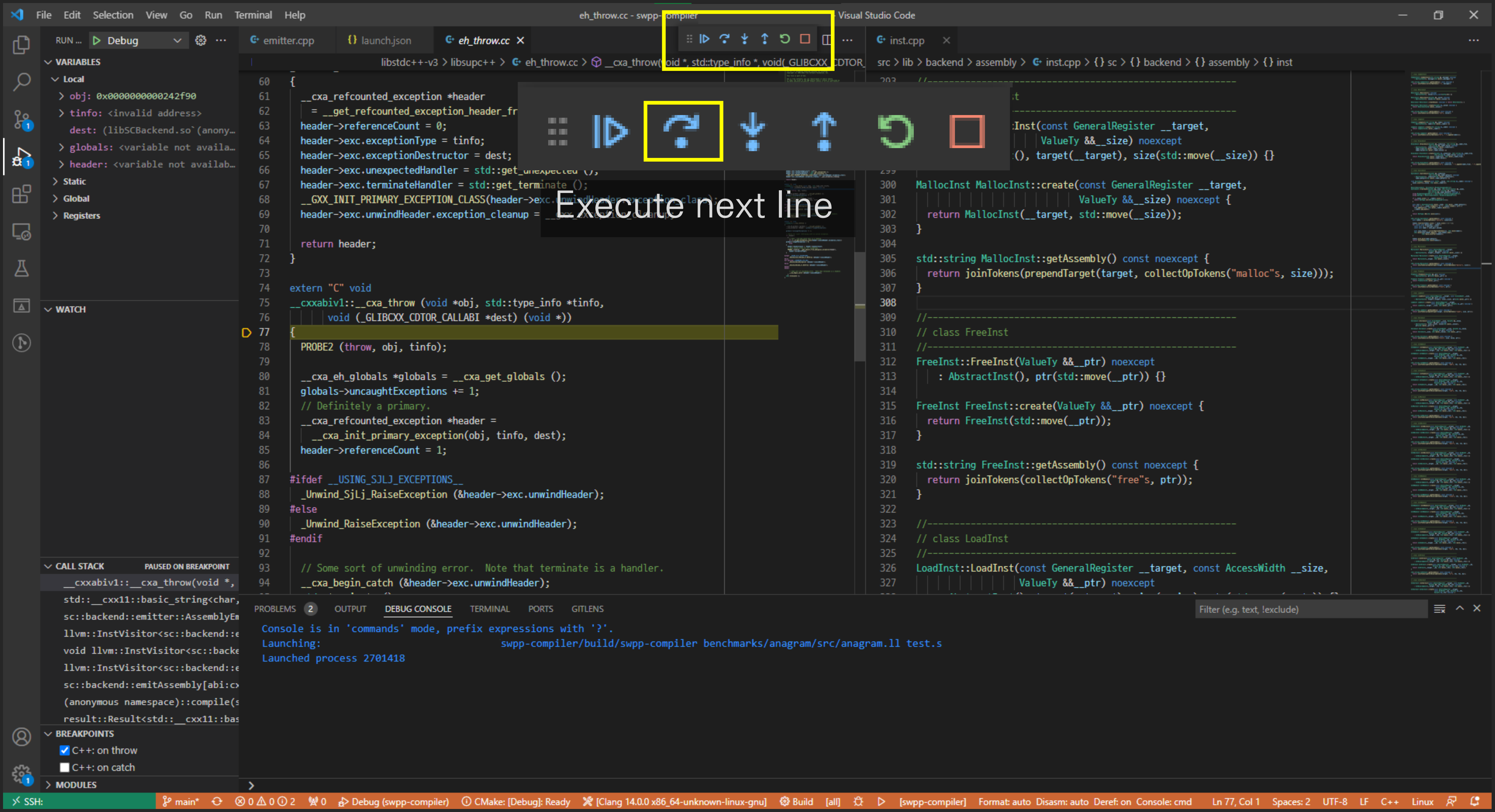
```
//-----------------------------------------------------------
t
Inst(const GeneralRegister __target,
                 ValueTy &&__size) noexcept
(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  nst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//-----------------------------------------------------------
// class FreeInst
//-----------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//-----------------------------------------------------------
// class LoadInst
//-----------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
__cxxabiv1::__cxa_throw(void *,
std::__cxx11::basic_string<char,
sc::backend::emitter::AssemblyEm
llvm::InstVisitor<sc::backend::e
void llvm::InstVisitor<sc::backe
llvm::InstVisitor<sc::backend::e
sc::backend::emitAssembly[abi:cx
(anonymous namespace)::compile(s
```

- [✓] C++: on throw
- [ ] C++: on catch

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                     swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

```
SSH:   main*   ⊗ 0 ⚠ 0 ⓘ 2   0   Debug (swpp-compiler)   CMake: [Debug]: Ready   [Clang 14.0.0 x86_64-unknown-linux-gnu]   Build   [all]   [swpp-compiler]   Format: auto  Disasm: auto  Deref: on  Console: cmd   Ln 77, Col 1   Spaces: 2   UTF-8   LF   C++   Linux
```


File Edit Selection View Go Run Terminal Help
eh_throw.cc - swpp-compiler - Visual Studio Code
RUN ... Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
Execute next line

```
VARIABLES
  Local
    > obj: 0x0000000000242f90
    > tinfo: <invalid address>
      dest: (libSCBackend.so`(anony…
    > globals: <variable not availa…
    > header: <variable not availab…
  > Static
  > Global
  > Registers
WATCH
CALL STACK                PAUSED ON BREAKPOINT
  __cxxabiv1::__cxa_throw(void *,
  std::__cxx11::basic_string<char,
  sc::backend::emitter::AssemblyEm
  llvm::InstVisitor<sc::backend::e
  void llvm::InstVisitor<sc::backe
  llvm::InstVisitor<sc::backend::e
  sc::backend::emitAssembly[abi:cx
  (anonymous namespace)::compile(s
  result::Result<std::__cxx11::bas
BREAKPOINTS
  [✓] C++: on throw
  [ ] C++: on catch
> MODULES
```

```
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_fr
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.
  header->exc.unwindHeader.exception_cleanup =

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
      void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  __cxa_eh_globals *globals = __cxa_get_globals ();
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

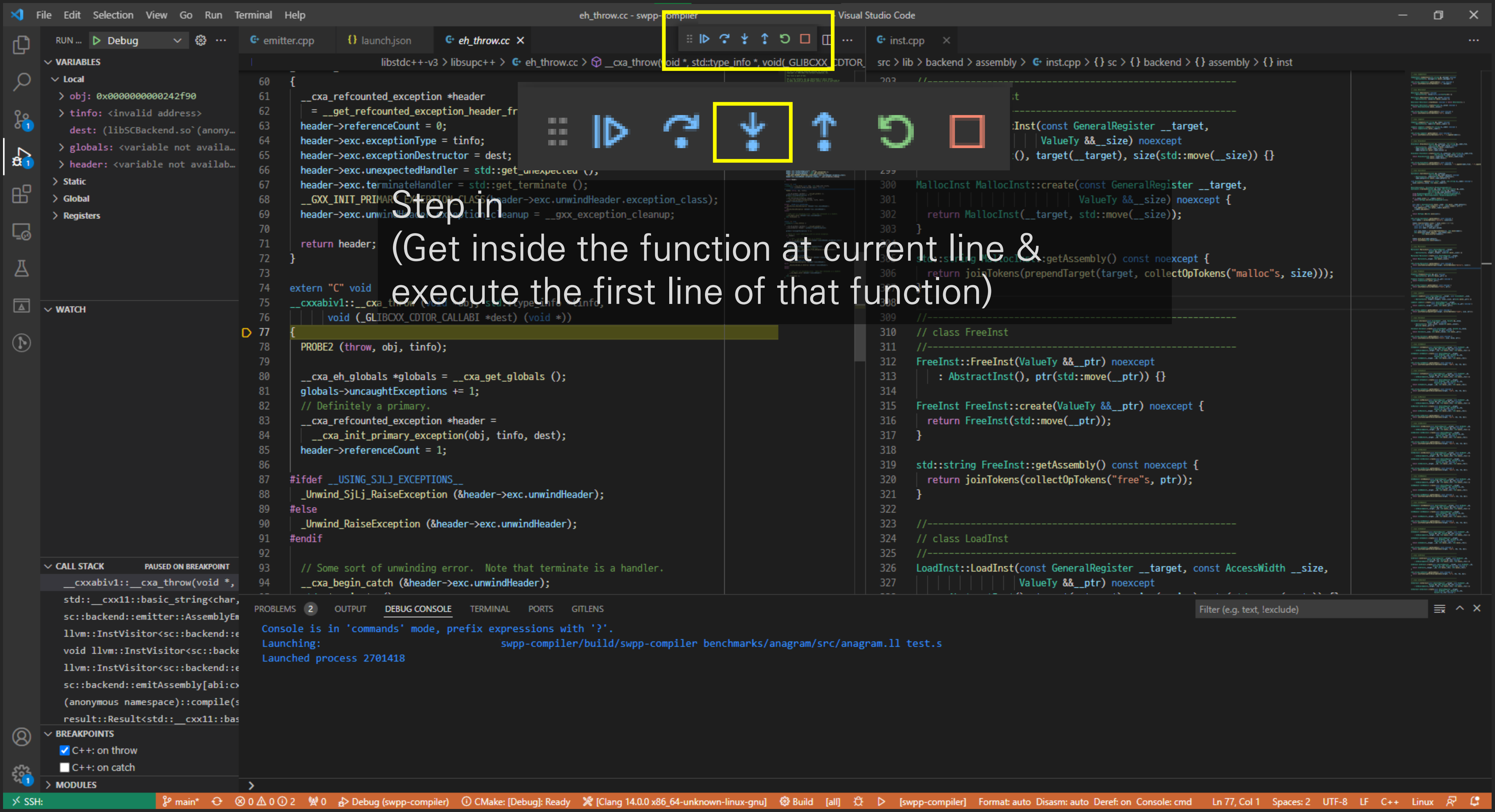
```
//--------------------------------------------------------------
                t
//--------------------------------------------------------------
                Inst(const GeneralRegister __target,
                     ValueTy &&__size) noexcept
                (), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//--------------------------------------------------------------
// class FreeInst
//--------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
  : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//--------------------------------------------------------------
// class LoadInst
//--------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                  swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

```
SSH:   main*   ⊗ 0 ⚠ 0 ⓘ 2   0   Debug (swpp-compiler)   CMake: [Debug]: Ready   [Clang 14.0.0 x86_64-unknown-linux-gnu]   Build   [all]   [swpp-compiler]   Format: auto  Disasm: auto  Deref: on  Console: cmd   Ln 77, Col 1   Spaces: 2   UTF-8   LF   C++   Linux
```

Step in
(Get inside the function at current line &
execute the first line of that function)



libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_

```
obj: 0x00000000000242f90
tinfo: <invalid address>
dest: (libSCBackend.so`(anony…
globals: <variable not availa…
header: <variable not availab…
```

```cpp
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_fr
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.unwindHeader.exception_class);
  header->exc.unwindHeader.exception_cleanup = __gxx_exception_cleanup;

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
      void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  __cxa_eh_globals *globals = __cxa_get_globals ();
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

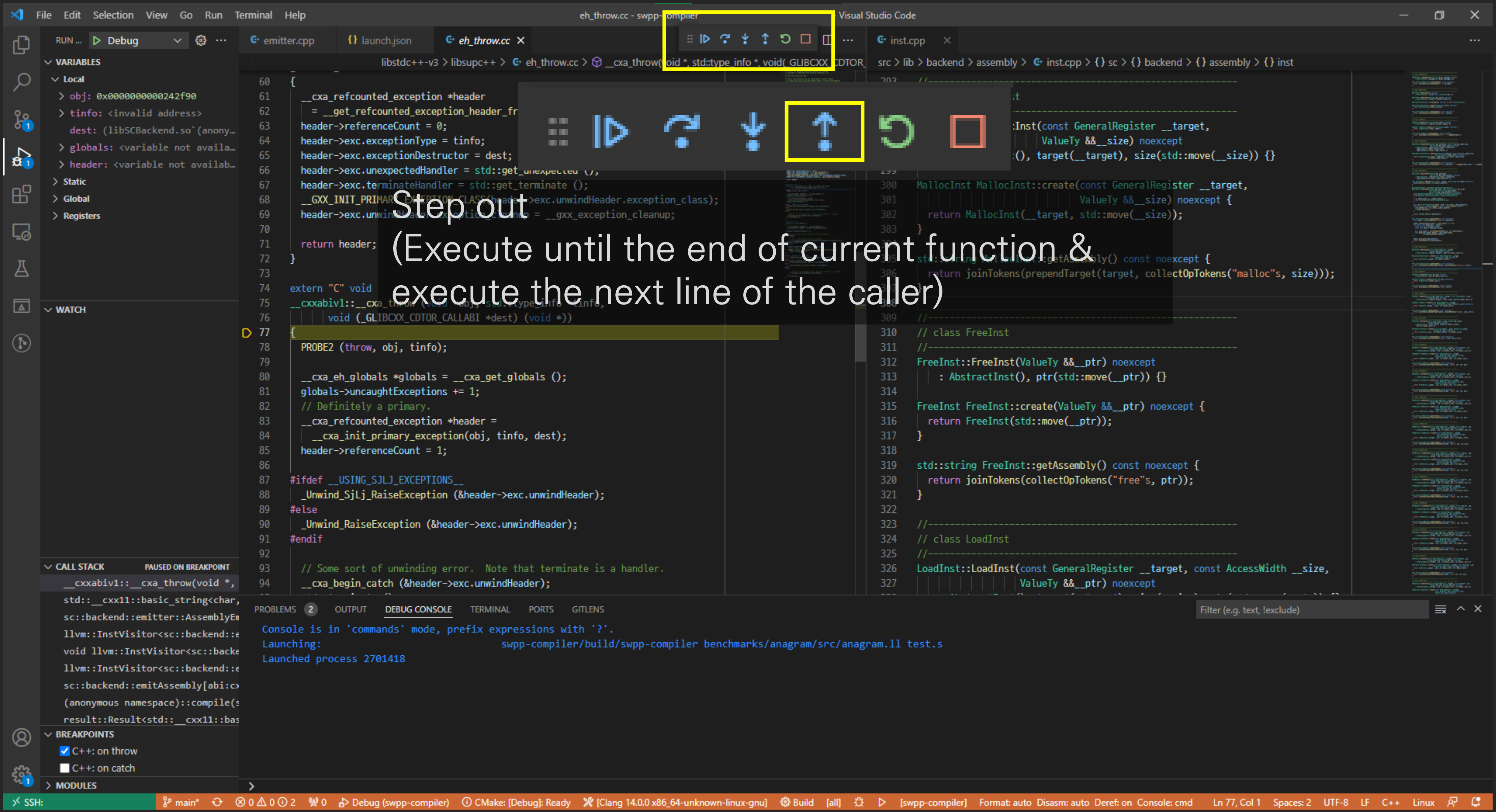
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst

```cpp
//------------------------------------------------------------
t
//------------------------------------------------------------
Inst(const GeneralRegister __target,
         ValueTy &&__size) noexcept
(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));

//------------------------------------------------------------
// class FreeInst
//------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
  : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//------------------------------------------------------------
// class LoadInst
//------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
__cxxabiv1::__cxa_throw(void *,
std::__cxx11::basic_string<char,
sc::backend::emitter::AssemblyEm
llvm::InstVisitor<sc::backend::e
void llvm::InstVisitor<sc::backe
llvm::InstVisitor<sc::backend::e
sc::backend::emitAssembly[abi:cx
(anonymous namespace)::compile(s
result::Result<std::__cxx11::bas
```

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                    swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

SSH: main* ⊗ 0 ⚠ 0 ⓘ 2 0 Debug (swpp-compiler) CMake: [Debug]: Ready [Clang 14.0.0 x86_64-unknown-linux-gnu] Build [all] [swpp-compiler] Format: auto Disasm: auto Deref: on Console: cmd Ln 77, Col 1 Spaces: 2 UTF-8 LF C++ Linux



Step out
(Execute until the end of current function & execute the next line of the caller)

```
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_fr
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.unwindHeader.exception_class);
  header->exc.unwindHeader.exception_cleanup = __gxx_exception_cleanup;

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
      void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  __cxa_eh_globals *globals = __cxa_get_globals ();
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

```
Inst(const GeneralRegister __target,
     ValueTy &&__size) noexcept
(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
  return MallocInst(__target, std::move(__size));
}

  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//----------------------------------------------------------
// class FreeInst
//----------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//----------------------------------------------------------
// class LoadInst
//----------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

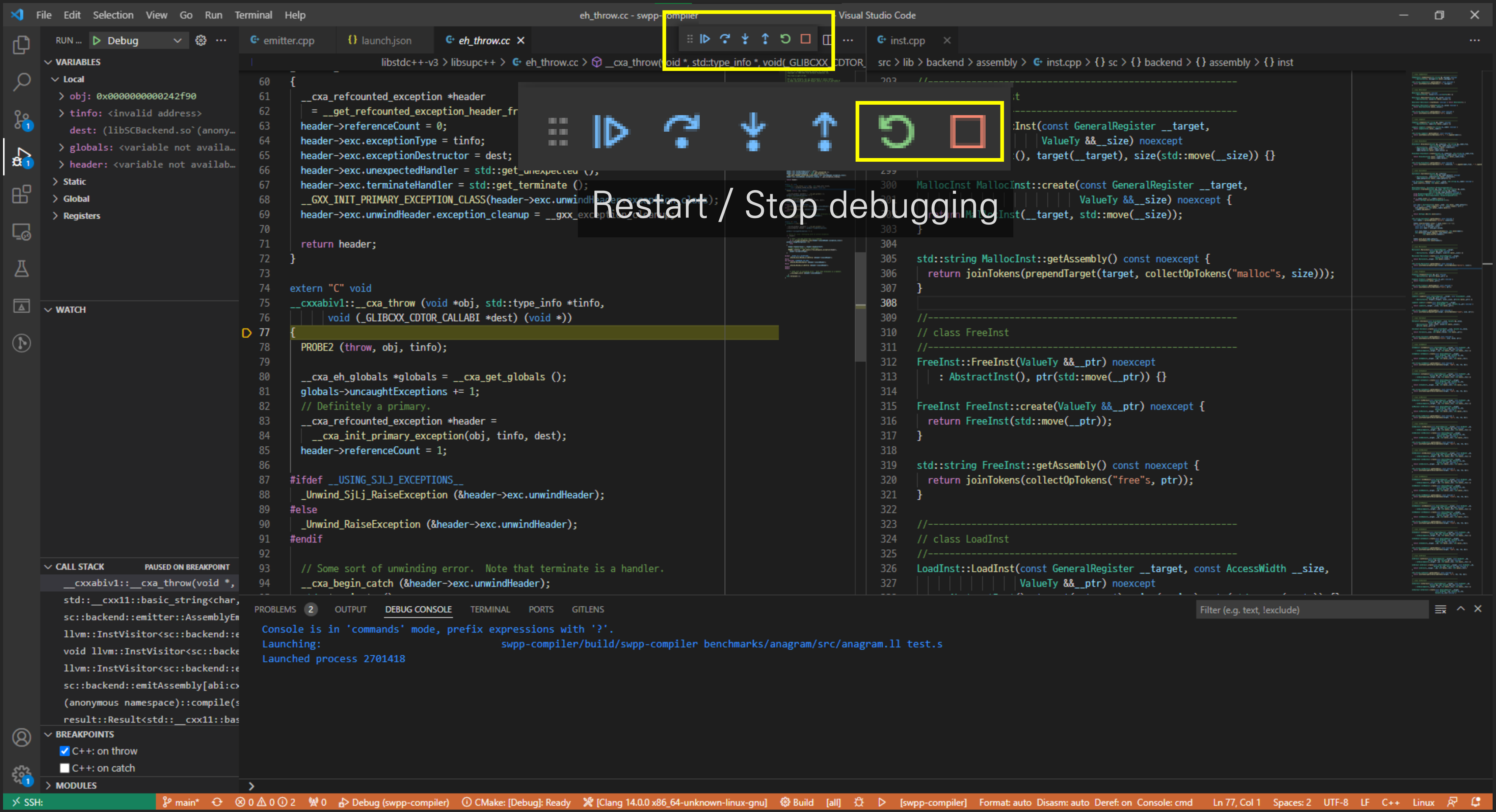
VARIABLES
Local
obj: 0x0000000000242f90
tinfo: <invalid address>
dest: (libSCBackend.so`(anony…
globals: <variable not availa…
header: <variable not availab…
Static
Global
Registers

CALL STACK — PAUSED ON BREAKPOINT
```
__cxxabiv1::__cxa_throw(void *,
std::__cxx11::basic_string<char,
sc::backend::emitter::AssemblyEm
llvm::InstVisitor<sc::backend::e
void llvm::InstVisitor<sc::backe
llvm::InstVisitor<sc::backend::e
sc::backend::emitAssembly[abi:cx
(anonymous namespace)::compile(s
result::Result<std::__cxx11::bas
```

BREAKPOINTS
☑ C++: on throw
☐ C++: on catch

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

File
Edit
Selection
View
Go
Run
Terminal
Help
eh_throw.cc - swpp-compiler - Visual Studio Code
RUN ...
Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
WATCH
MODULES
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude)
SSH:
main*
Debug (swpp-compiler)
CMake: [Debug]: Ready
[Clang 14.0.0 x86_64-unknown-linux-gnu]
Build
[all]
[swpp-compiler]
Format: auto Disasm: auto Deref: on Console: cmd
Ln 77, Col 1
Spaces: 2
UTF-8
LF
C++
Linux


eh_throw.cc - swpp-compiler - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN ... Debug
emitter.cpp
launch.json
eh_throw.cc
inst.cpp
libstdc++-v3 > libsupc++ > eh_throw.cc > __cxa_throw(void *, std::type_info *, void(_GLIBCXX_CDTOR_
src > lib > backend > assembly > inst.cpp > {} sc > {} backend > {} assembly > {} inst
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Filter (e.g. text, !exclude)

```
VARIABLES
  Local
    > obj: 0x0000000000242f90
    > tinfo: <invalid address>
      dest: (libSCBackend.so`(anony…
    > globals: <variable not availa…
    > header: <variable not availab…
  > Static
  > Global
  > Registers

WATCH

CALL STACK                     PAUSED ON BREAKPOINT
  __cxxabiv1::__cxa_throw(void *,
  std::__cxx11::basic_string<char,
  sc::backend::emitter::AssemblyEm
  llvm::InstVisitor<sc::backend::e
  void llvm::InstVisitor<sc::backe
  llvm::InstVisitor<sc::backend::e
  sc::backend::emitAssembly[abi:cx
  (anonymous namespace)::compile(s
  result::Result<std::__cxx11::bas

BREAKPOINTS
  [✓] C++: on throw
  [ ] C++: on catch
MODULES
```

```cpp
{
  __cxa_refcounted_exception *header
    = __get_refcounted_exception_header_fr
  header->referenceCount = 0;
  header->exc.exceptionType = tinfo;
  header->exc.exceptionDestructor = dest;
  header->exc.unexpectedHandler = std::get_unexpected ();
  header->exc.terminateHandler = std::get_terminate ();
  __GXX_INIT_PRIMARY_EXCEPTION_CLASS(header->exc.unwindH
  header->exc.unwindHeader.exception_cleanup = __gxx_exc

  return header;
}

extern "C" void
__cxxabiv1::__cxa_throw (void *obj, std::type_info *tinfo,
                         void (_GLIBCXX_CDTOR_CALLABI *dest) (void *))
{
  PROBE2 (throw, obj, tinfo);

  __cxa_eh_globals *globals = __cxa_get_globals ();
  globals->uncaughtExceptions += 1;
  // Definitely a primary.
  __cxa_refcounted_exception *header =
    __cxa_init_primary_exception(obj, tinfo, dest);
  header->referenceCount = 1;

#ifdef __USING_SJLJ_EXCEPTIONS__
  _Unwind_SjLj_RaiseException (&header->exc.unwindHeader);
#else
  _Unwind_RaiseException (&header->exc.unwindHeader);
#endif

  // Some sort of unwinding error.  Note that terminate is a handler.
  __cxa_begin_catch (&header->exc.unwindHeader);
```

```cpp
//--------------------------------------------------------------
Inst(const GeneralRegister __target,
            ValueTy &&__size) noexcept
(), target(__target), size(std::move(__size)) {}

MallocInst MallocInst::create(const GeneralRegister __target,
                              ValueTy &&__size) noexcept {
Inst(__target, std::move(__size));
}

std::string MallocInst::getAssembly() const noexcept {
  return joinTokens(prependTarget(target, collectOpTokens("malloc"s, size)));
}

//--------------------------------------------------------------
// class FreeInst
//--------------------------------------------------------------
FreeInst::FreeInst(ValueTy &&__ptr) noexcept
    : AbstractInst(), ptr(std::move(__ptr)) {}

FreeInst FreeInst::create(ValueTy &&__ptr) noexcept {
  return FreeInst(std::move(__ptr));
}

std::string FreeInst::getAssembly() const noexcept {
  return joinTokens(collectOpTokens("free"s, ptr));
}

//--------------------------------------------------------------
// class LoadInst
//--------------------------------------------------------------
LoadInst::LoadInst(const GeneralRegister __target, const AccessWidth __size,
                   ValueTy &&__ptr) noexcept
```

```
Console is in 'commands' mode, prefix expressions with '?'.
Launching:                    swpp-compiler/build/swpp-compiler benchmarks/anagram/src/anagram.ll test.s
Launched process 2701418
```

```
SSH:   main*   0 ⚠ 0 ⓘ 2   0   Debug (swpp-compiler)   CMake: [Debug]: Ready   [Clang 14.0.0 x86_64-unknown-linux-gnu]   Build   [all]   [swpp-compiler]   Format: auto  Disasm: auto  Deref: on  Console: cmd   Ln 77, Col 1   Spaces: 2   UTF-8   LF   C++   Linux
```

Restart / Stop debugging

Narrowing Down with Debugger

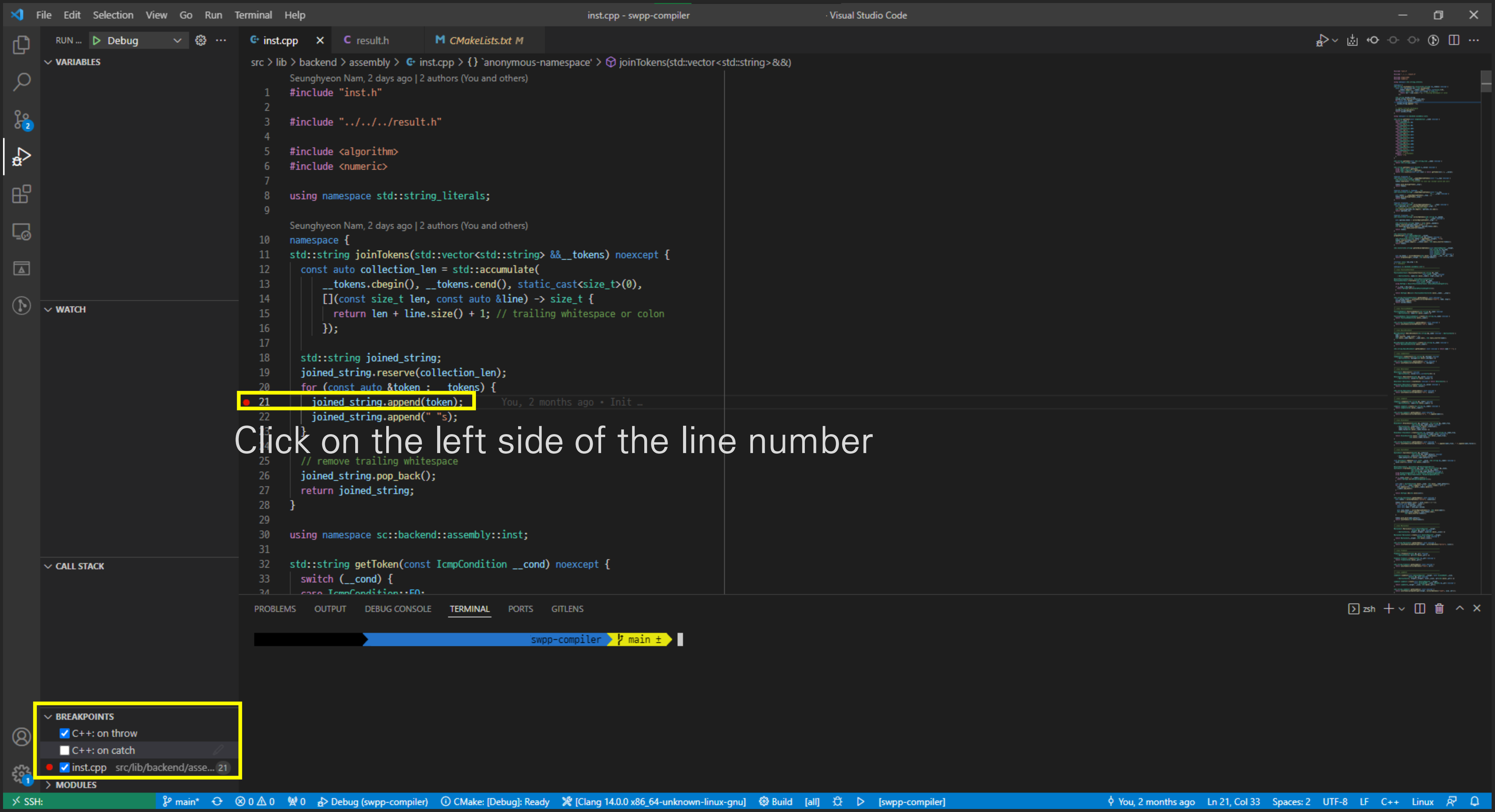
- How do you locate the exact line that crashes?
  - Debugger pinpoints (automatically pauses on) the crash site
  - No need to insert new code, rebuild, etc.

Traceback with Debugger

- You have to locate the code that first went wrong
    - Debugger shows you the call stack at the moment of the crash
    - Clicking is all you need to navigate through the call stacks
    - Find the first call stack with unexpected value or control flow

Traceback with Debugger

- Sometimes you have to monitor the change of values
  - If your code does not throw or crash, debugger won’t pause
  - You can use breakpoint to pause execution at a certain point



Click on the left side of the line number

inst.cpp - swpp-compiler - Visual Studio Code

src > lib > backend > assembly > inst.cpp > {} `anonymous-namespace' > joinTokens(std::vector<std::string>&&)

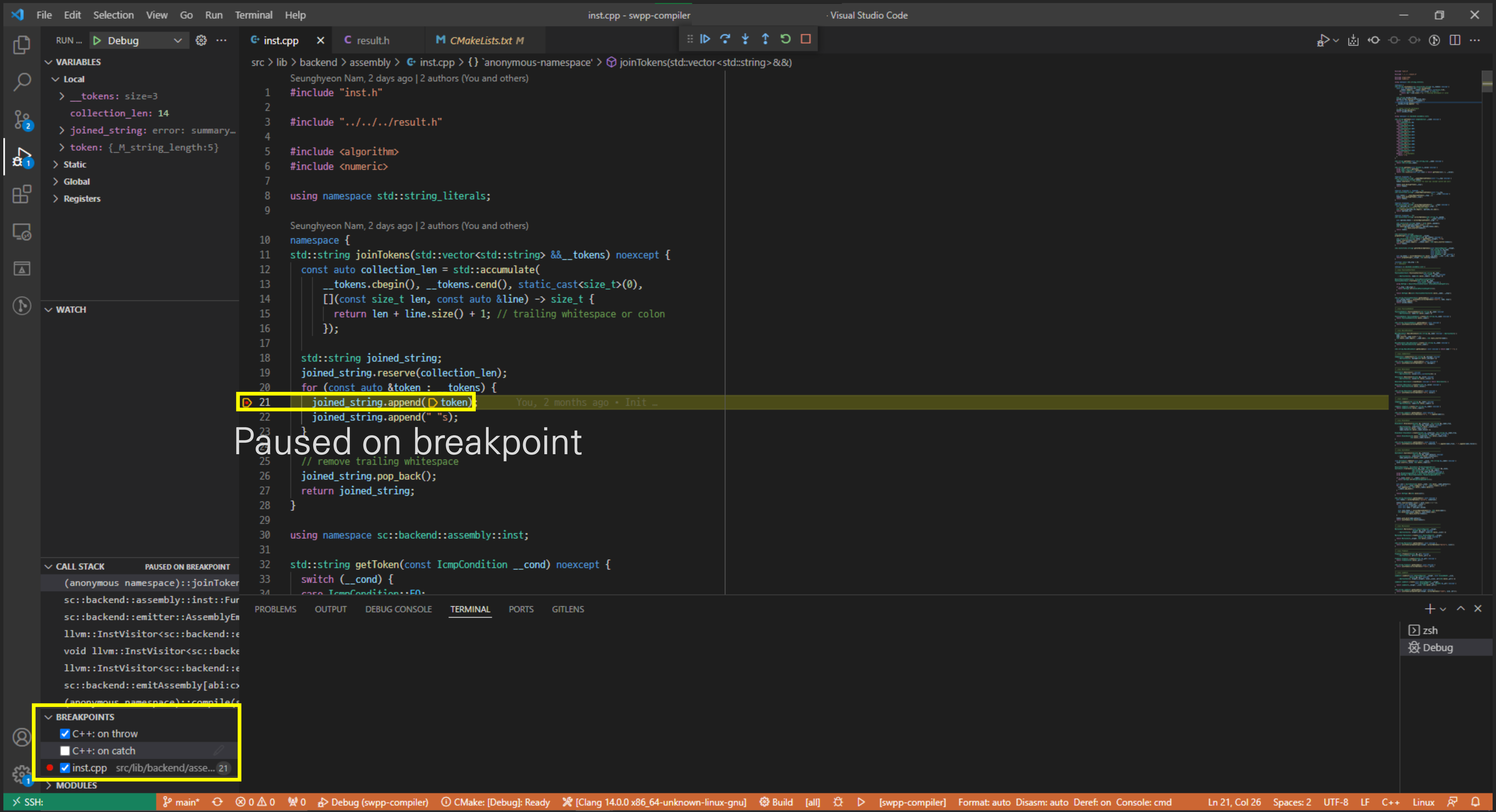
```cpp
Seunghyeon Nam, 2 days ago | 2 authors (You and others)
#include "inst.h"

#include "../../../result.h"

#include <algorithm>
#include <numeric>

using namespace std::string_literals;

Seunghyeon Nam, 2 days ago | 2 authors (You and others)
namespace {
std::string joinTokens(std::vector<std::string> &&__tokens) noexcept {
  const auto collection_len = std::accumulate(
      __tokens.cbegin(), __tokens.cend(), static_cast<size_t>(0),
      [](const size_t len, const auto &line) -> size_t {
        return len + line.size() + 1; // trailing whitespace or colon
      });

  std::string joined_string;
  joined_string.reserve(collection_len);
  for (const auto &token : __tokens) {
    joined_string.append(token);          You, 2 months ago • Init …
    joined_string.append(" "s);
  }

  // remove trailing whitespace
  joined_string.pop_back();
  return joined_string;
}

using namespace sc::backend::assembly::inst;

std::string getToken(const IcmpCondition __cond) noexcept {
  switch (__cond) {
```

BREAKPOINTS
- [x] C++: on throw
- [ ] C++: on catch
- [x] inst.cpp src/lib/backend/asse... 21

```
swpp-compiler  main ±
```

SSH: main*   0 ⚠ 0   Debug (swpp-compiler)   CMake: [Debug]: Ready   [Clang 14.0.0 x86_64-unknown-linux-gnu]   Build   [all]   [swpp-compiler]   You, 2 months ago   Ln 21, Col 33   Spaces: 2   UTF-8   LF   C++   Linux

Paused on breakpoint


File Edit Selection View Go Run Terminal Help
inst.cpp - swpp-compiler · Visual Studio Code
RUN ... Debug
inst.cpp
result.h
CMakeLists.txt M
src > lib > backend > assembly > inst.cpp > {} `anonymous-namespace' > joinTokens(std::vector<std::string>&&)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
zsh
Debug

```
VARIABLES
  Local
    > __tokens: size=3
      collection_len: 14
    > joined_string: error: summary…
    > token: {_M_string_length:5}
  > Static
  > Global
  > Registers
WATCH
```

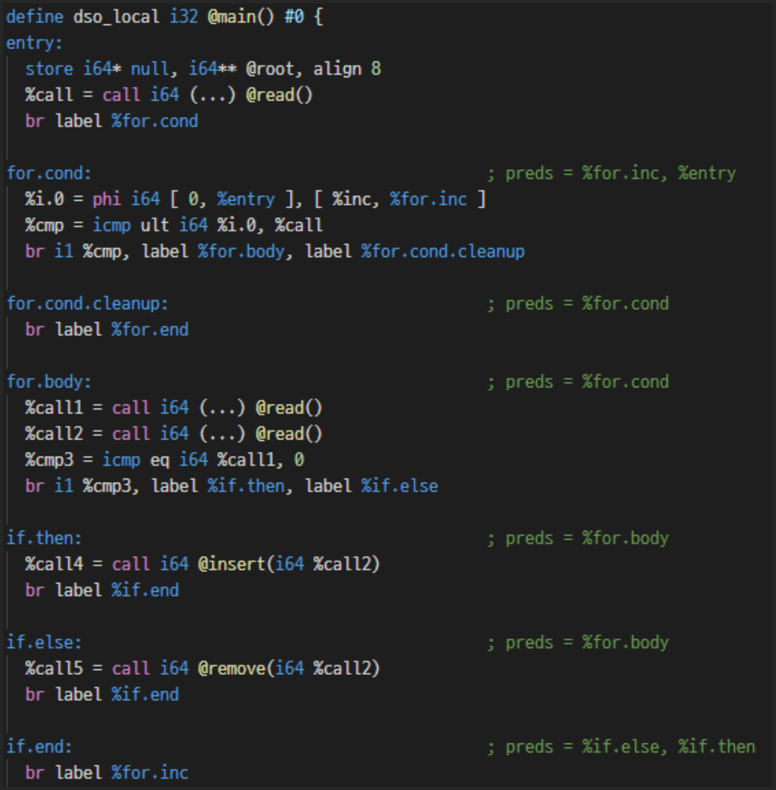
```
Seunghyeon Nam, 2 days ago | 2 authors (You and others)
#include "inst.h"

#include "../../../result.h"

#include <algorithm>
#include <numeric>

using namespace std::string_literals;

Seunghyeon Nam, 2 days ago | 2 authors (You and others)
namespace {
std::string joinTokens(std::vector<std::string> &&__tokens) noexcept {
  const auto collection_len = std::accumulate(
      __tokens.cbegin(), __tokens.cend(), static_cast<size_t>(0),
      [](const size_t len, const auto &line) -> size_t {
        return len + line.size() + 1; // trailing whitespace or colon
      });

  std::string joined_string;
  joined_string.reserve(collection_len);
  for (const auto &token : __tokens) {
    joined_string.append(token);      You, 2 months ago • Init …
    joined_string.append(" "s);
  }

  // remove trailing whitespace
  joined_string.pop_back();
  return joined_string;
}

using namespace sc::backend::assembly::inst;

std::string getToken(const IcmpCondition __cond) noexcept {
  switch (__cond) {
  case IcmpCondition::EQ:
```

```
CALL STACK                PAUSED ON BREAKPOINT
  (anonymous namespace)::joinToker
  sc::backend::assembly::inst::Fur
  sc::backend::emitter::AssemblyEm
  llvm::InstVisitor<sc::backend::e
  void llvm::InstVisitor<sc::backe
  llvm::InstVisitor<sc::backend::e
  sc::backend::emitAssembly[abi:cx
  (anonymous namespace)::compile(s
BREAKPOINTS
  ☑ C++: on throw
  ☐ C++: on catch
  ● ☑ inst.cpp  src/lib/backend/asse…  21
MODULES
```

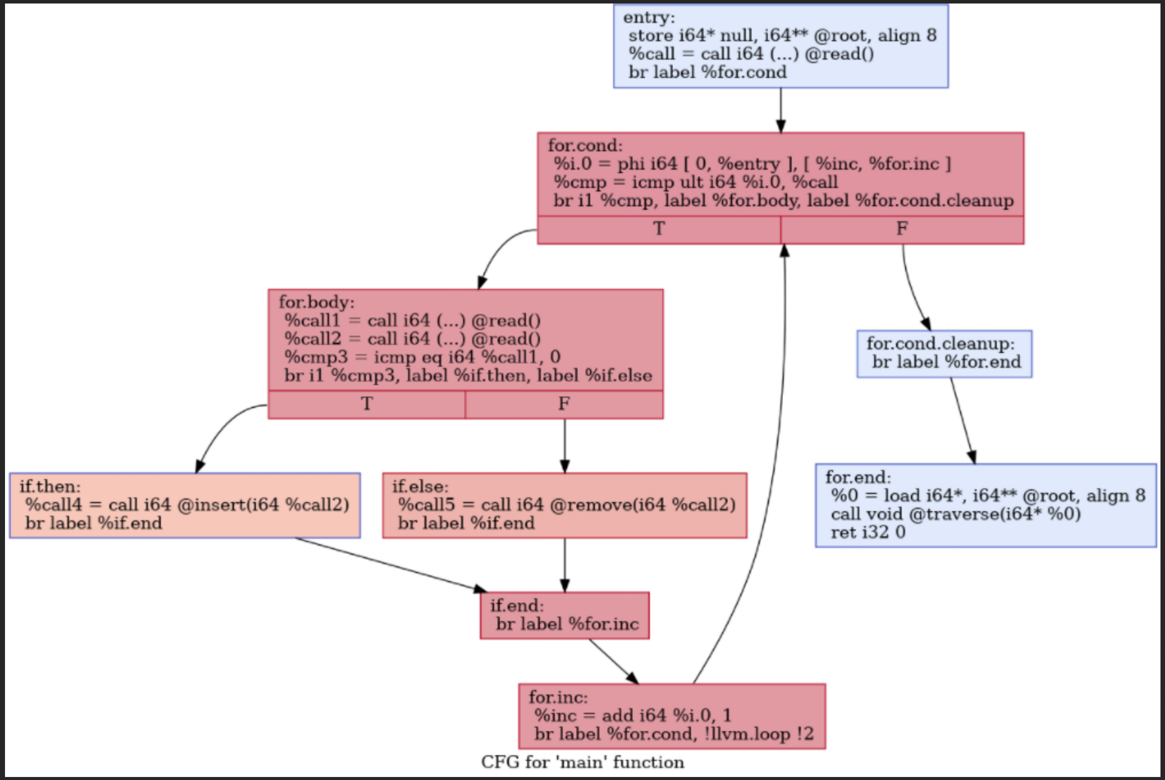
```
SSH:  main*  ⊗ 0 ⚠ 0  0  Debug (swpp-compiler)  CMake: [Debug]: Ready  [Clang 14.0.0 x86_64-unknown-linux-gnu]  Build  [all]  [swpp-compiler]  Format: auto  Disasm: auto  Deref: on  Console: cmd  Ln 21, Col 26  Spaces: 2  UTF-8  LF  C++  Linux
```

IR Visualization

- Visualizing the control flow of your IR program can be helpful



```
define dso_local i32 @main() #0 {
entry:
  store i64* null, i64** @root, align 8
  %call = call i64 (...) @read()
  br label %for.cond

for.cond:                                         ; preds = %for.inc, %entry
  %i.0 = phi i64 [ 0, %entry ], [ %inc, %for.inc ]
  %cmp = icmp ult i64 %i.0, %call
  br i1 %cmp, label %for.body, label %for.cond.cleanup

for.cond.cleanup:                                 ; preds = %for.cond
  br label %for.end

for.body:                                         ; preds = %for.cond
  %call1 = call i64 (...) @read()
  %call2 = call i64 (...) @read()
  %cmp3 = icmp eq i64 %call1, 0
  br i1 %cmp3, label %if.then, label %if.else

if.then:                                          ; preds = %for.body
  %call4 = call i64 @insert(i64 %call2)
  br label %if.end

if.else:                                          ; preds = %for.body
  %call5 = call i64 @remove(i64 %call2)
  br label %if.end

if.end:                                           ; preds = %if.else, %if.then
  br label %for.inc
```



```
entry:
 store i64* null, i64** @root, align 8
 %call = call i64 (...) @read()
 br label %for.cond

for.cond:
 %i.0 = phi i64 [ 0, %entry ], [ %inc, %for.inc ]
 %cmp = icmp ult i64 %i.0, %call
 br i1 %cmp, label %for.body, label %for.cond.cleanup
 T | F

for.body:
 %call1 = call i64 (...) @read()
 %call2 = call i64 (...) @read()
 %cmp3 = icmp eq i64 %call1, 0
 br i1 %cmp3, label %if.then, label %if.else
 T | F

for.cond.cleanup:
 br label %for.end

if.then:
 %call4 = call i64 @insert(i64 %call2)
 br label %if.end

if.else:
 %call5 = call i64 @remove(i64 %call2)
 br label %if.end

for.end:
 %0 = load i64*, i64** @root, align 8
 call void @traverse(i64* %0)
 ret i32 0

if.end:
 br label %for.inc

for.inc:
 %inc = add i64 %i.0, 1
 br label %for.cond, !llvm.loop !2
```

CFG for 'main' function

IR Visualization

- Install GraphViz
  - Use the package manager to handle dependencies for you
- Run `<llvm-dir>/opt --dot-cfg <IR-program.ll>`
  - You'll get a .dot file for each function in the program
- Run `dot <dot-file.dot> -Tpng -o <image-name.png>`