





int main() {



println(*S);

```
auto shared_data = shared_ptr<mutex>::make_shared("Hello there");
```

for(int i: num_thread)



string^s=lock_guard^.begin();

^>append(("🔥"));

```
void entry_point(shared_ptr<mutex<string>> data, int thread_id) {
```

ved to ir< theads theads{ };

auto_lock_guard = data->lock();

```
thead.push_back(thread(&entry_processor, shared_data, i));
```

const int n = 15;



apply(tid) %>% auto %>%

```
void identify_point(std::shared_ptr<synchronized_val<std::string>> sync_s, int id)
```

s.append(("🔥"));

irevivis: 5



```
auto s = std::make_shared<sync_hrnsized_val<std::string>>("Hello there");
```

*SYNOPSIS:



```
std::vector<safe_thread> threads {};
```



const int num_threads = 15;

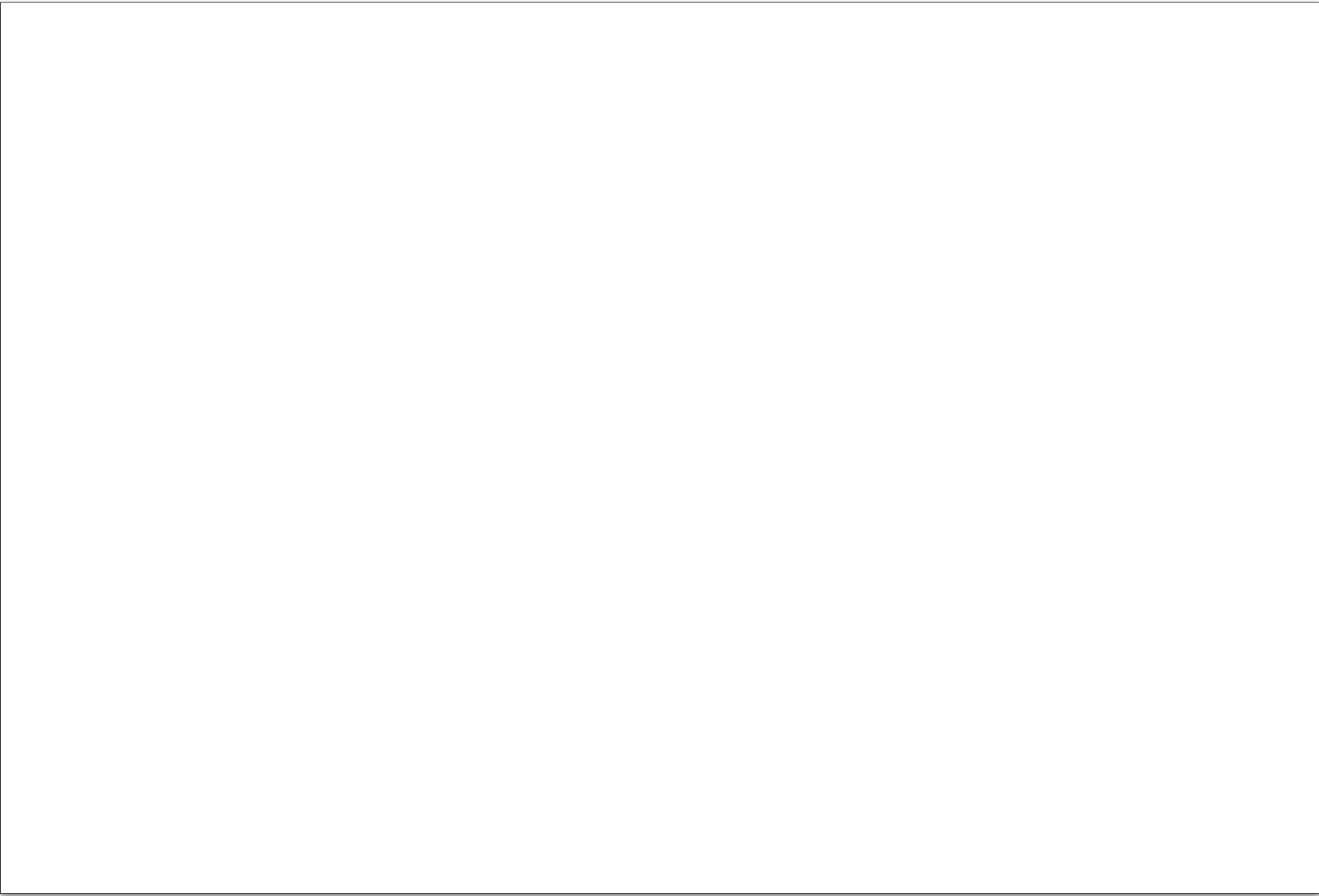
it's

main ()

```
std::println("{} {}" , tid);
```

```
thead.push_back(safe_thead(entry_ptr, auto(i)));
```

```
for(int i:std::iota(0, num_threads))
```

copy(shared_data, i));

shared_ptr<mutex> ring_data,



int main() {
 return 0;
}

`int thread_id)` safe

^>append("🔥");





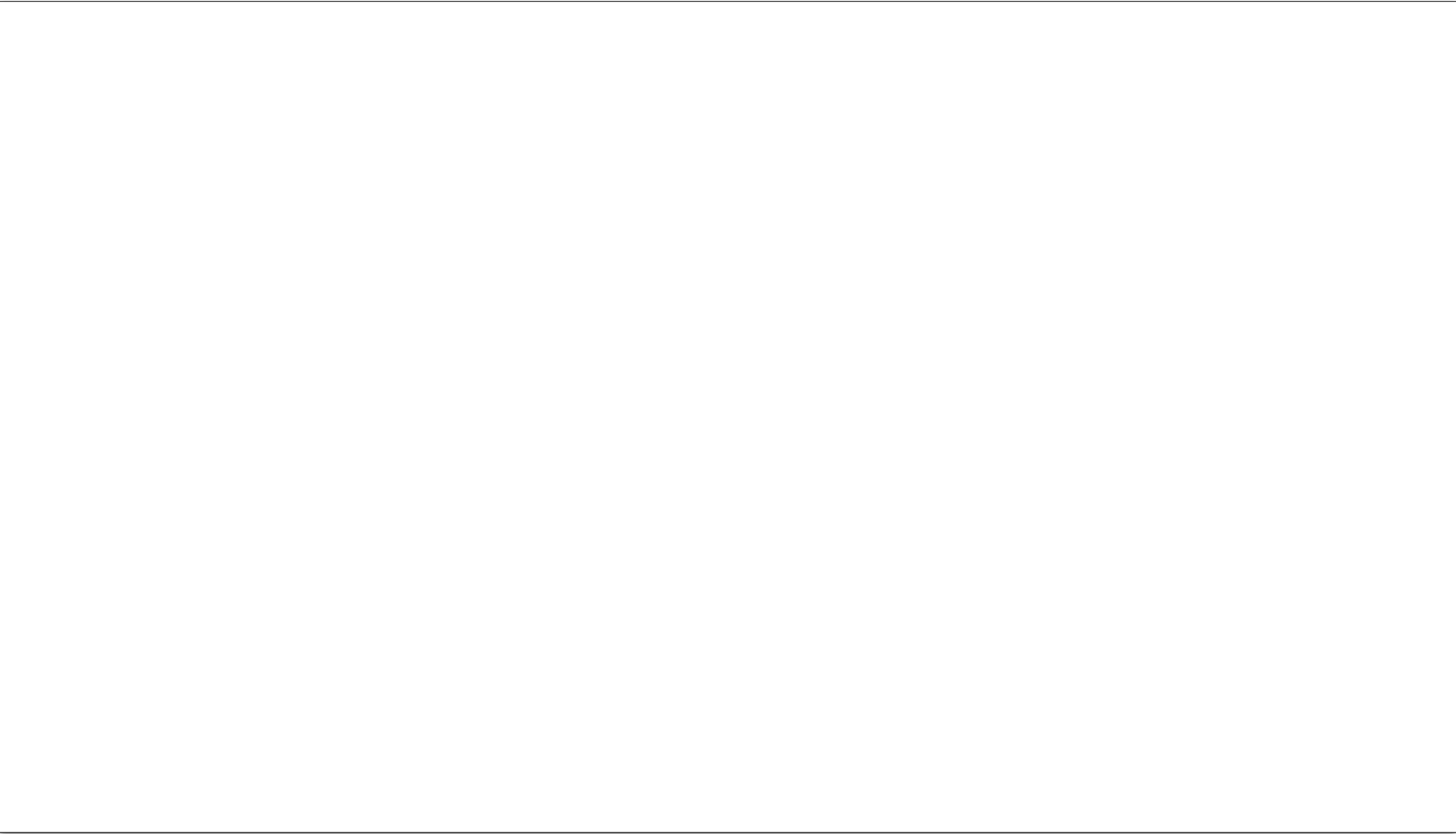
thead^push_back(thead(&entry_ptr,

void entity_propoint(

```
auto lock_guard=>lock();
```

println(*s);

string^s==lock_guard^._borrow();



void entity_print(

s.append("🔥");

int

tid)

apply(tid) (autos) }

```
std::shared_ptr<synchro_ni_zed_valued> data,
```



```
std::println("{ }", s, tid);
```

*data)

Int main()

netturns

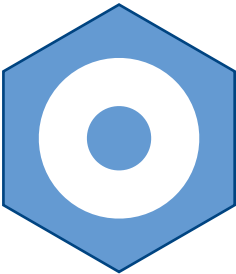
auto(s), auto(i);

threads.push_back(safe_thread(entry_ptr,











102

