Genode

для C++ разработчиков

Сергей Платонов

C++ Russia 2021

План

В чём проблема?

Микроядро

Компоненты

Создание компонента

Создание сервиса

Взаимодействие

Disclaimer

Disclaimer

не разработчик ОС

Disclaimer

не разработчик ОС

применение: embedded

Disclaimer

не разработчик ОС

применение: embedded

сравнение с linux

Disclaimer

не разработчик ОС

применение: embedded

сравнение с linux

это не введение в Genode

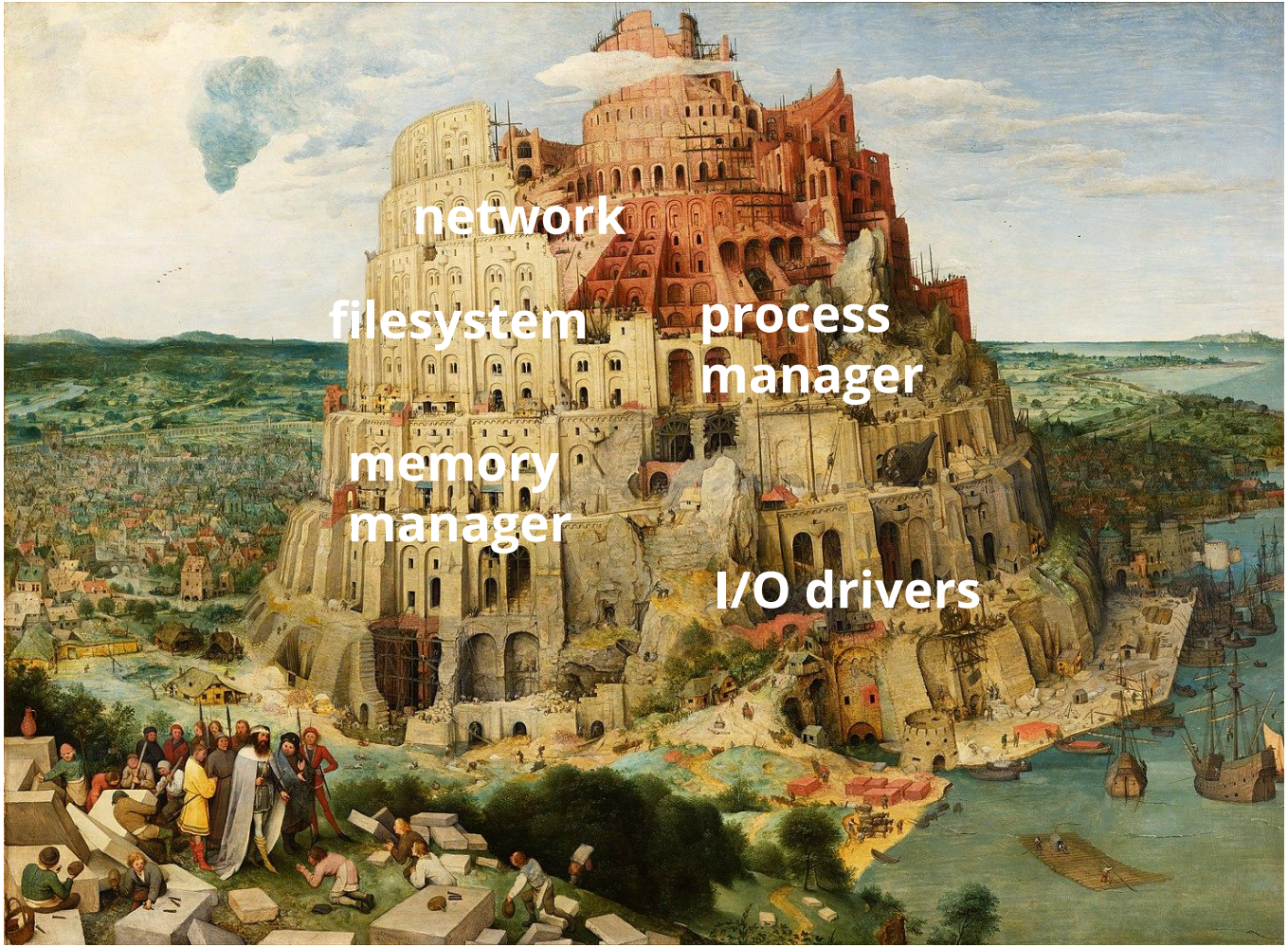
Цель доклада

знакомство с Genode

создание компонента

создание сервиса

их взаиомдействие


network
filesystem
process manager
memory manager
I/O drivers

Genode

Genode

микроядро

Genode

микроядро

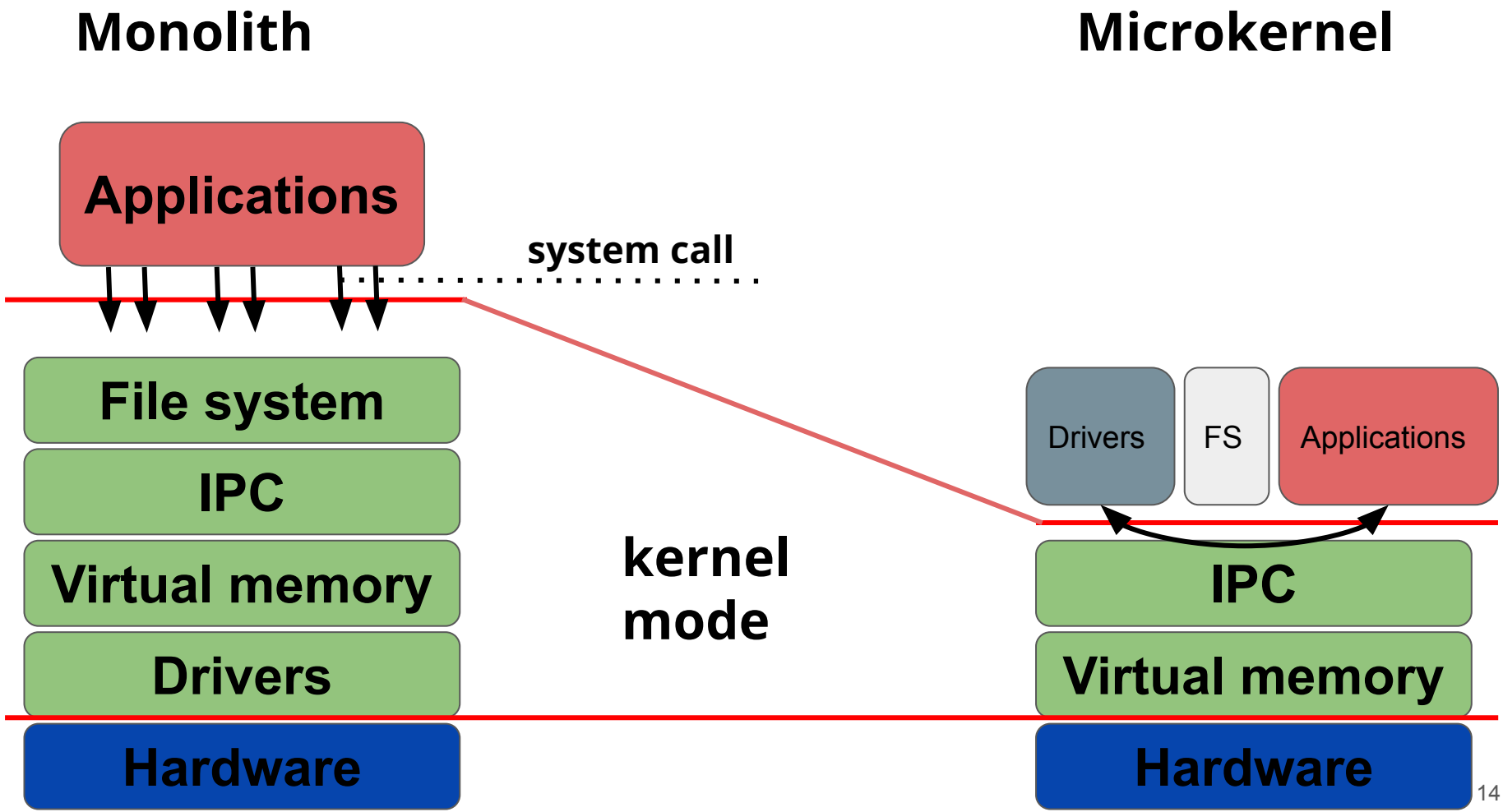
capability-based модель безопасности

Genode

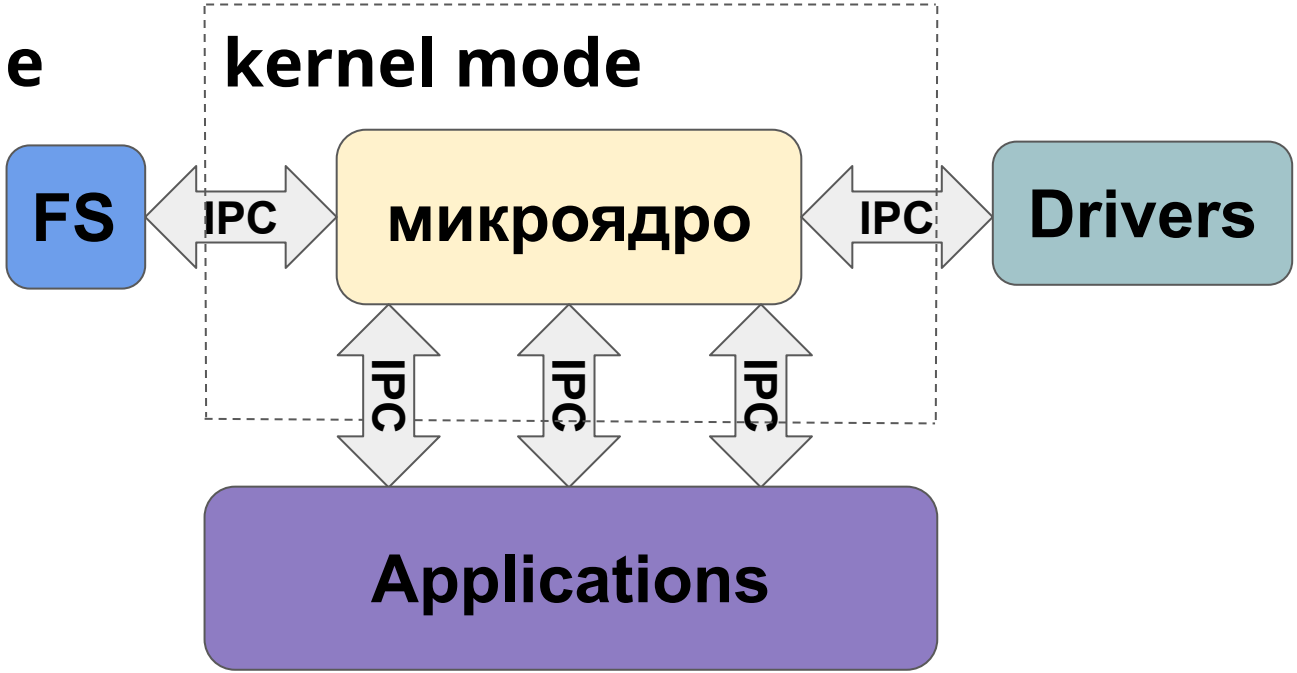
микроядро

capability-based модель безопасности

объём доверенного кода (trusted code base)


Monolith
Microkernel
Applications
system call
File system
IPC
Virtual memory
Drivers
Hardware
kernel mode
Drivers
FS
Applications
IPC
Virtual memory
Hardware

Микроядро


user mode
kernel mode
FS
IPC
микроядро
IPC
Drivers
IPC
IPC
IPC
Applications

Genode: доступные ядра

nova

sel4

L4/Fiasco

base-hw

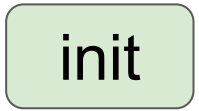
linux

Genode: структура

компонент — базовый строительный блок

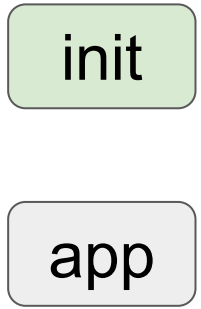
Genode: структура

компонент — базовый строительный блок



Genode: структура

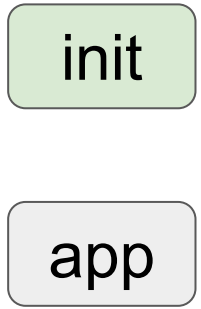
компонент — базовый строительный блок



Genode: структура

компонент — базовый строительный блок

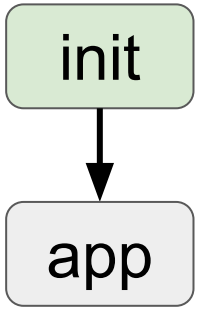
отношение “родитель — потомок”



Genode: структура

компонент — базовый строительный блок

отношение “родитель — потомок”

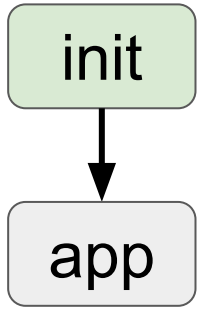


Genode: структура

компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов



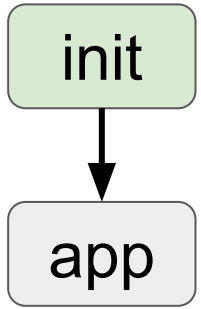
Genode: структура

компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов

сервисы



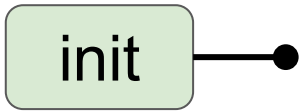
Genode: структура

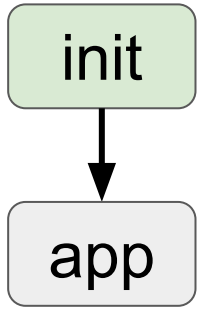
компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов

сервисы





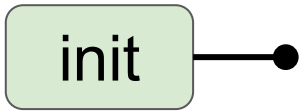
Genode: структура

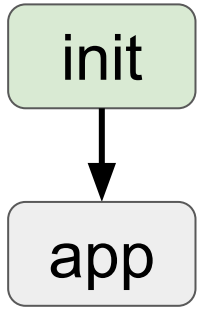
компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов

сервисы и сессии





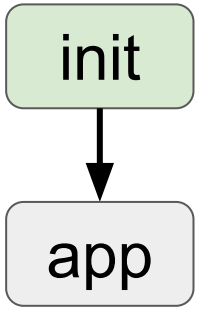
Genode: структура

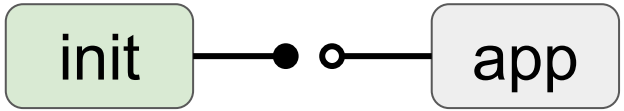
компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов

сервисы и сессии





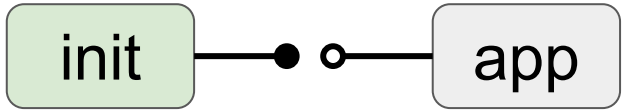
Genode: структура

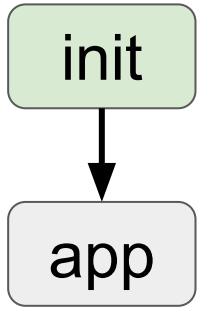
компонент — базовый строительный блок

отношение “родитель — потомок”

дерево компонентов

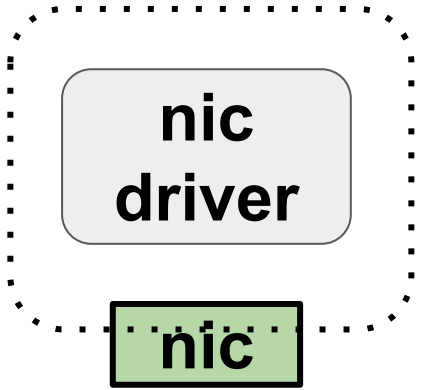
сервисы и сессии



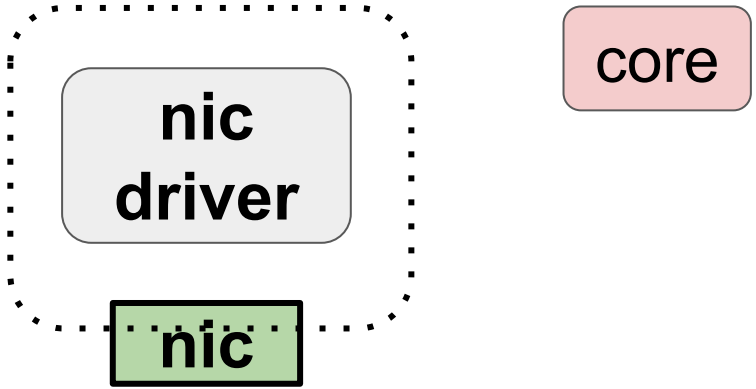


у каждого компонента свой protection domain

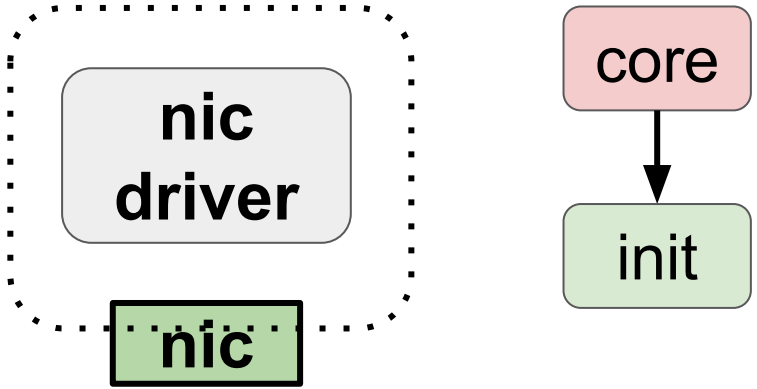
Genode: сервисы



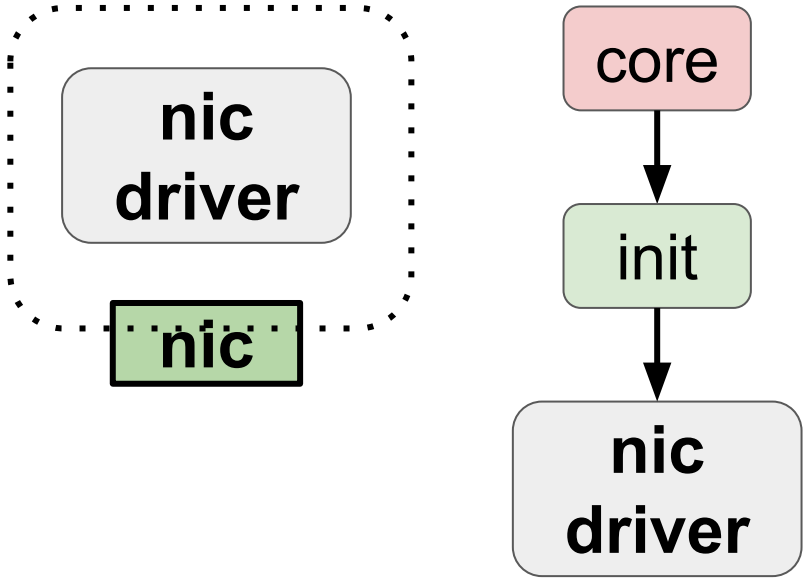
Genode: сервисы



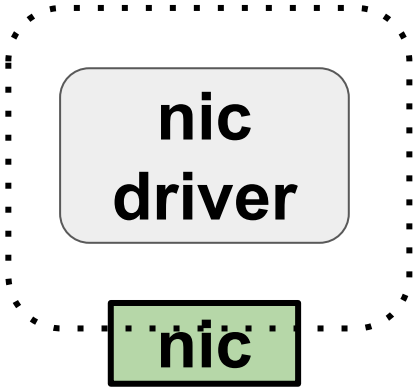
Genode: сервисы

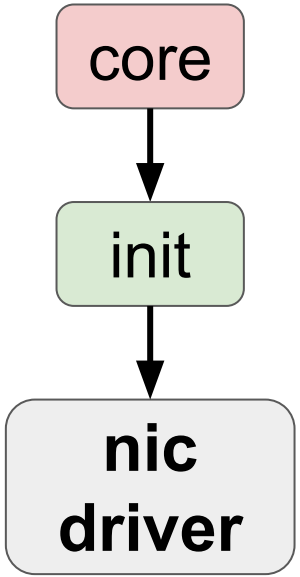


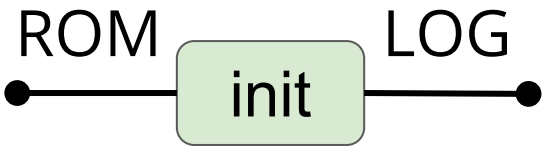
Genode: сервисы



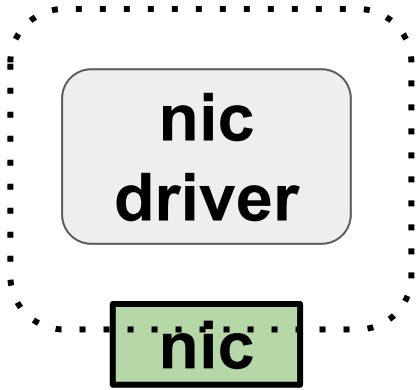
Genode: сервисы

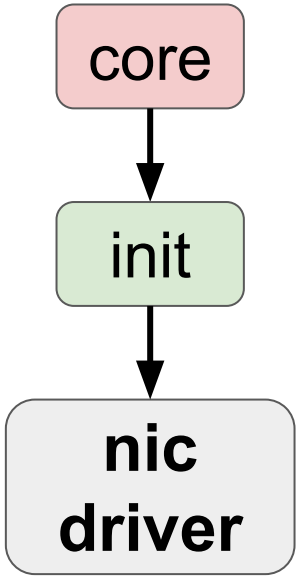


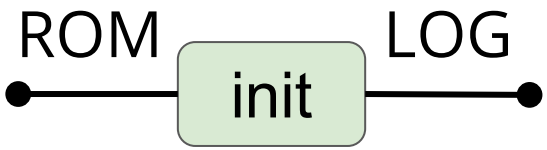




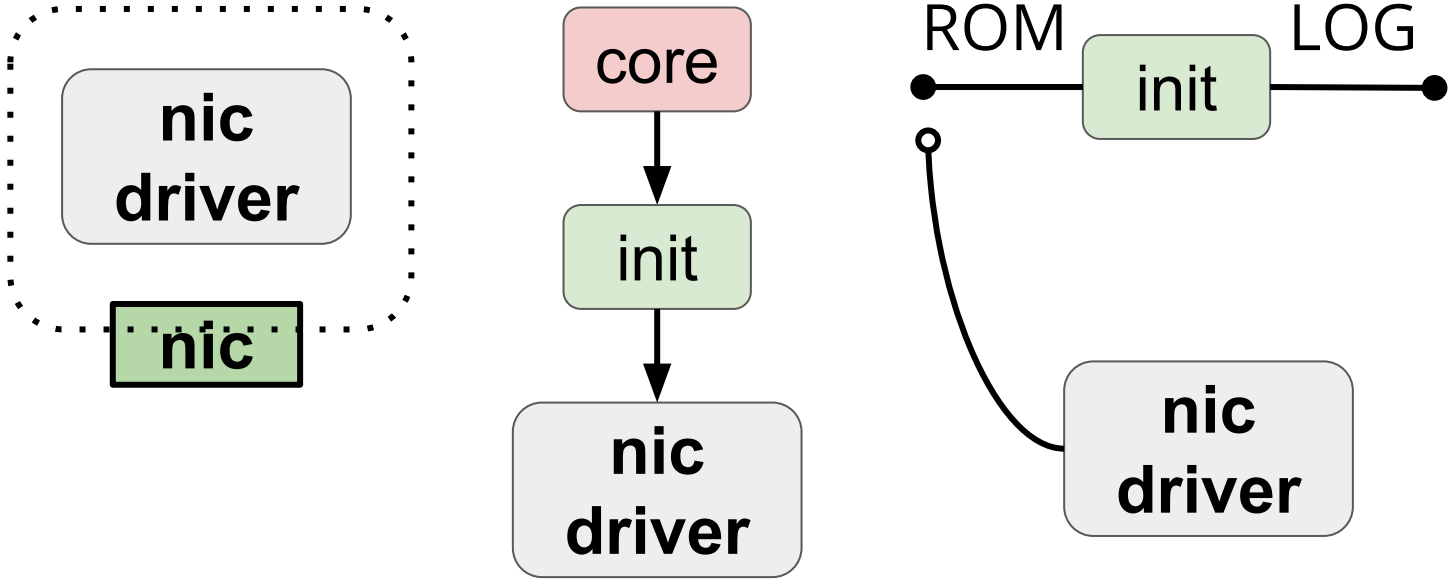
Genode: сервисы



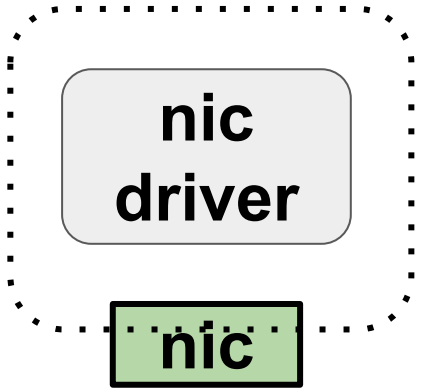


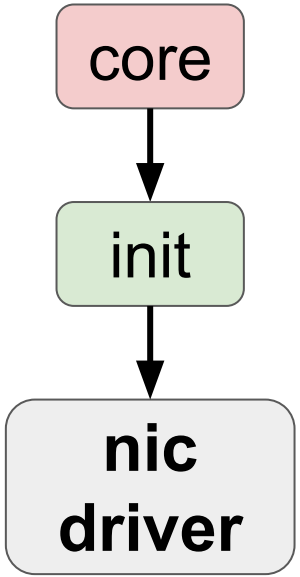


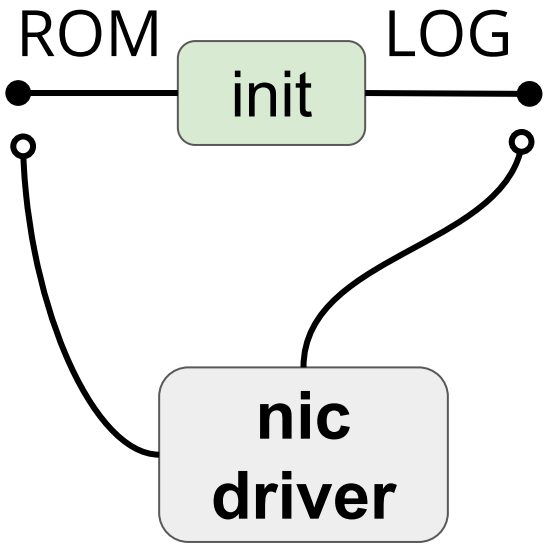
Genode: сервисы



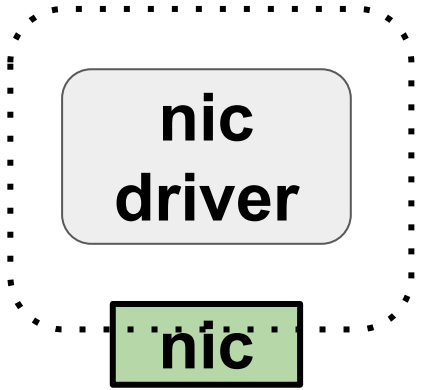
Genode: сервисы

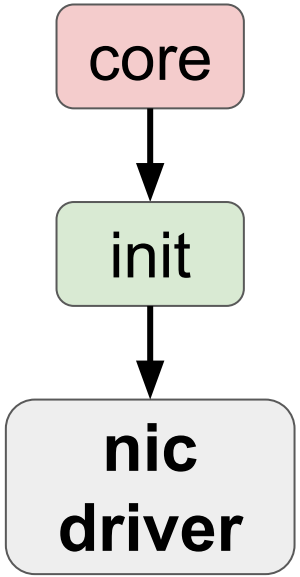


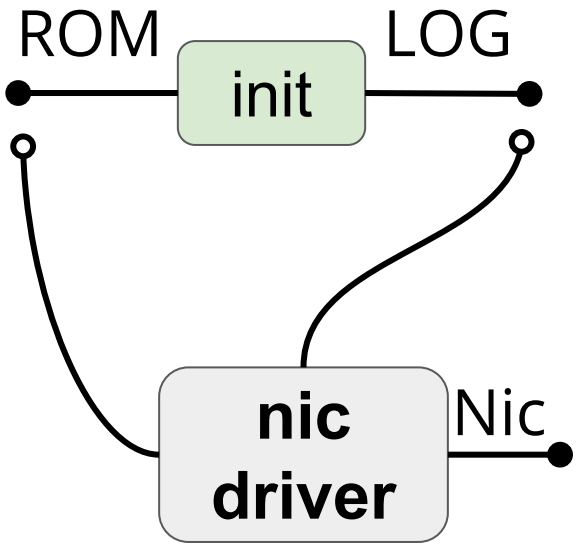




Genode: сервисы







Компонент log_time

Компонент log_time

раз в секунду выводит текущую дату и время

Компонент log_time

раз в секунду выводит текущую дату и время

стандартный C++

Создаём компонент log_time

```
$ mkdir -p src/app/log_timer run
```

Создаём компонент log_time

```
$ mkdir -p src/app/log_timer run

$ touch src/app/log_timer/main.cc \
        src/app/log_timer/target.mk
```

Создаём компонент log_time

```
$ mkdir -p src/app/log_timer run

$ touch src/app/log_timer/main.cc \
        src/app/log_timer/target.mk

$ touch run/log_time.run
```

main.cc

```
1  int main() {
2      char buffer[255];
3      while (true) {
4          time_t now = to_time_t(now());
5          strftime(str, sizeof(str), "%F %T", localtime(&now));
6          cout << str << endl;
7          this_thread::sleep_for(1s);
8      }
9      return 0;
10 }
```

target.mk

```
1  TARGET = log_time
2  SRC_C  = main.cc
3  LIBS   = libc posix stdcxx
```

конфигурация init

```
1  <config>
2      <parent-provides>
3          <service name="ROM"/>
4          <service name="LOG"/>
5          <service name="CPU"/>
6          <service name="PD"/>
7      </parent-provides>
```

конфигурация init

```
1  <config>
2      <parent-provides>
3          <service name="ROM"/>
4          <service name="LOG"/>
5          <service name="CPU"/>
6          <service name="PD"/>
7      </parent-provides>
```

конфигурация init

```
1  <config>
2     <parent-provides>
3         <service name="ROM"/>
4         <service name="LOG"/>
5         <service name="CPU"/>
6         <service name="PD"/>
7     </parent-provides>
```

конфигурация init

```
1  <config>
2     <parent-provides>
3         <service name="ROM"/>
4         <service name="LOG"/>
5         <service name="CPU"/>
6         <service name="PD"/>
7     </parent-provides>
```

конфигурация init

```
1  <config>
2      <parent-provides>
3          <service name="ROM"/>
4          <service name="LOG"/>
5          <service name="CPU"/>
6          <service name="PD"/>
7      </parent-provides>
```

конфигурация init

```
1  <config>
2      <parent-provides>
3          <service name="ROM"/>
4          <service name="LOG"/>
5          <service name="CPU"/>
6          <service name="PD"/>
7      </parent-provides>
```

конфигурация init

```
 8  <start name="timer" caps="100">
 9     <resource name="RAM" quantum="1M"/>
10     <provides><service name="Timer"/></provides>
11     <route>
12         <any-service> <parent/> </any-service>
13     </route>
14  </start>
```

конфигурация init

```
 8  <start name="timer" caps="100">
 9      <resource name="RAM" quantum="1M"/>
10      <provides><service name="Timer"/></provides>
11      <route>
12          <any-service> <parent/> </any-service>
13      </route>
14  </start>
```

конфигурация init

```
 8  <start name="timer" caps="100">
 9      <resource name="RAM" quantum="1M"/>
10      <provides><service name="Timer"/></provides>
11      <route>
12          <any-service> <parent/> </any-service>
13      </route>
14  </start>
```

конфигурация init

```
 8  <start name="timer" caps="100">
 9      <resource name="RAM" quantum="1M"/>
10      <provides><service name="Timer"/></provides>
11      <route>
12          <any-service> <parent/> </any-service>
13      </route>
14  </start>
```

конфигурация init

```
 8  <start name="timer" caps="100">
 9      <resource name="RAM" quantum="1M"/>
10      <provides><service name="Timer"/></provides>
11      <route>
12          <any-service> <parent/> </any-service>
13      </route>
14  </start>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
15  <start name="log_time" caps="200">
16      <resource name="RAM" quantum="4M"/>
17      <config>
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
```

конфигурация init

```
24      <route>
25          <service name="Timer">
26              <child name="timer"/>
27          </service>
28          <any-service> <parent/> </any-service>
29      </route>
30  </start>
31  </config>
```

конфигурация init

```
24      <route>
25          <service name="Timer">
26              <child name="timer"/>
27          </service>
28          <any-service> <parent/> </any-service>
29      </route>
30  </start>
31  </config>
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so rtc_timer stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so rtc_timer stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so rtc_timer stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so rtc_timer stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

log_time.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    posix.lib.so rtc_timer stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

Готовимся к запуску

Готовимся к запуску

```
$ ln -s `pwd` ~/genode/repos/cpp-russia-2021
```

Готовимся к запуску

```
$ ln -s `pwd` ~/genode/repose/cpp-russia-2021
```

```
$ ~/genode/tool/create_builddir x86_64
```

Готовимся к запуску

```
$ ln -s `pwd` ~/genode/repose/cpp-russia-2021

$ ~/genode/tool/create_builddir x86_64

$ for repo in cpp-russia-2021 gems ports; do
echo ”REPOSITORIES += \$(GENODE_DIR)/repos/$repo” >>
  ~/genode/build/x86_64/etc/build.conf
done
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init] Warning: log_time: no route to service "Rtc"
(label="log_time -> ")
[init -> log_time] Error: Rtc-session creation failed
(ram_quota=8K, label="")
[init -> log_time] Error: failed to create <rtc> VFS node
[init -> log_time] Warning: /dev/rtc not readable, returning 0
[init -> log_time] 1970-01-01 00:00:00
```

конфигурация init

```
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
24      <route>
25          <service name="Timer">
26               <child name="timer"/>
27          </service>
```

конфигурация init

```
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
24      <route>
25          <service name="Timer">
26              <child name="timer"/>
27          </service>
```

конфигурация init

```
18          <libc stdin="/dev/null" stdout="/dev/log"
19                stderr="/dev/log" rtc="/dev/rtc"/>
20          <vfs>
21              <dir name="dev"><log/><null/><rtc/></dir>
22          </vfs>
23      </config>
24      <route>
25          <service name="Timer">
26              <child name="timer"/>
27          </service>
```

log_time.run

```
21      <dir name="dev">
22           <log/><null/>
23           <inline name=”rtc”>2021-11-10 00:01</inline>
24      </dir>
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init -> log_time] 2021-11-10 00:01:00
[init -> log_time] 2021-11-10 00:01:01
[init -> log_time] 2021-11-10 00:01:02
[init -> log_time] 2021-11-10 00:01:03
...
```

Как тестировать log_time?

Создаём компонент mock_rtc

```
$ mkdir -p src/test/mock_rtc
```

Создаём компонент mock_rtc

```
$ mkdir -p src/test/mock_rtc

$ touch src/test/mock_rtc/main.cc \
        src/test/mock_rtc/target.mk
```

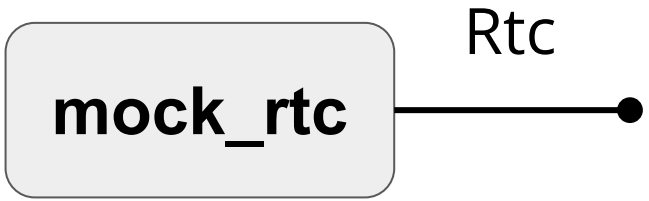
Создаём компонент mock_rtc

```
$ mkdir -p src/test/mock_rtc

$ touch src/test/mock_rtc/main.cc \
        src/test/mock_rtc/target.mk
```

mock_rtc

Создаём компонент mock_rtc

```
$ mkdir -p src/test/mock_rtc

$ touch src/test/mock_rtc/main.cc \
        src/test/mock_rtc/target.mk
```



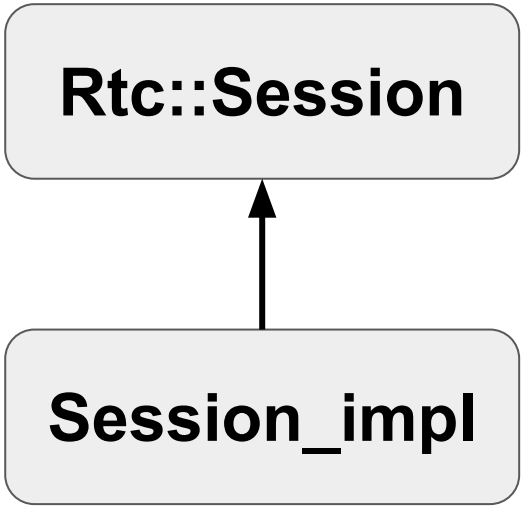
Создаём компонент mock_rtc

```
$ mkdir -p src/test/mock_rtc

$ touch src/test/mock_rtc/main.cc \
        src/test/mock_rtc/target.mk
```

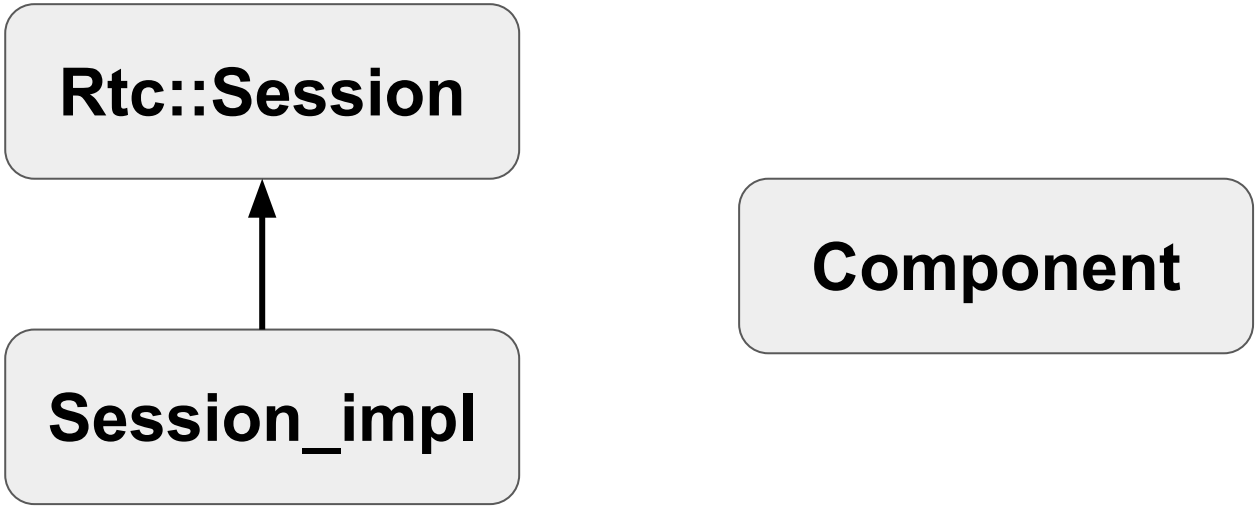
Структура компонента mock_rtc

Rtc::Session

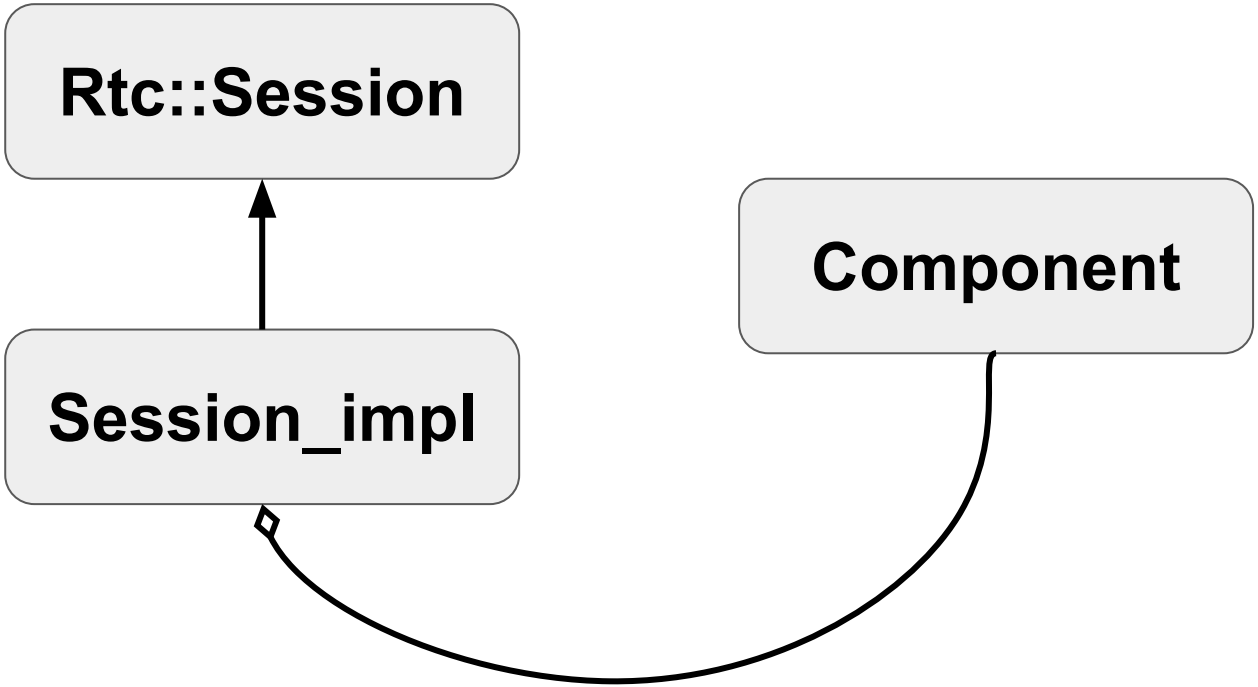
Структура компонента mock_rtc



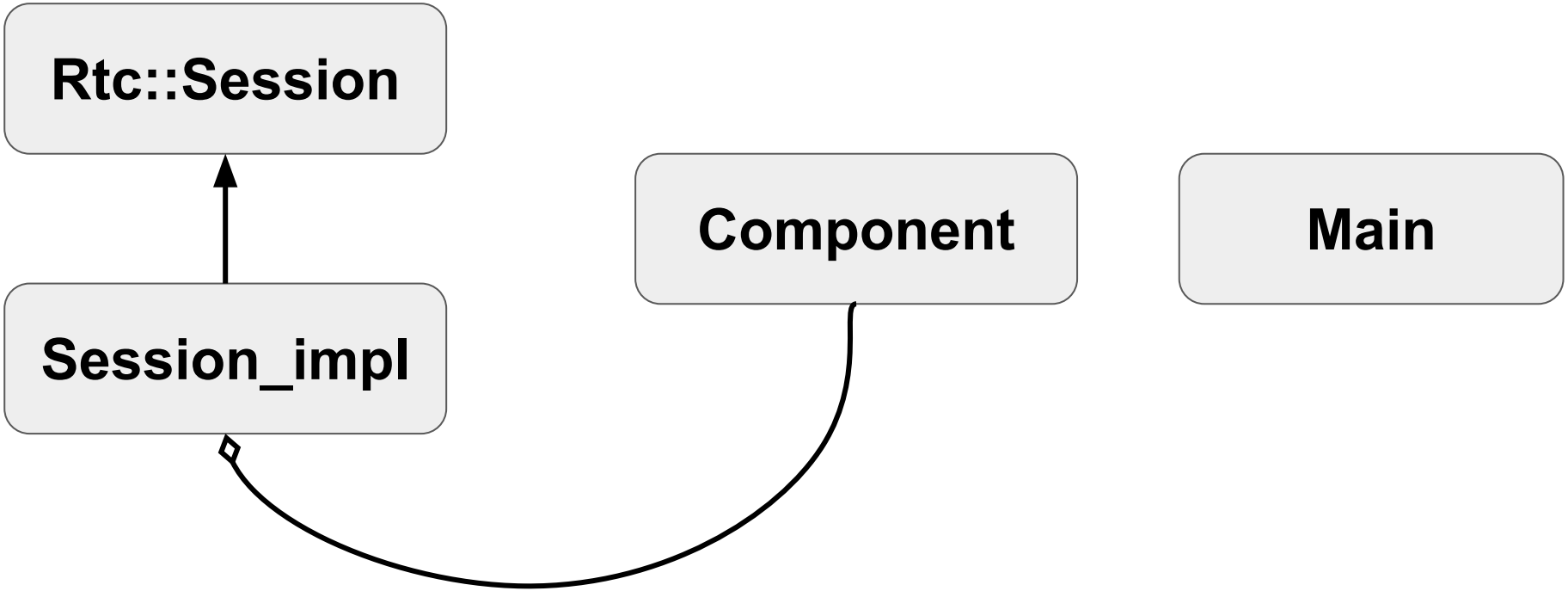
Структура компонента mock_rtc



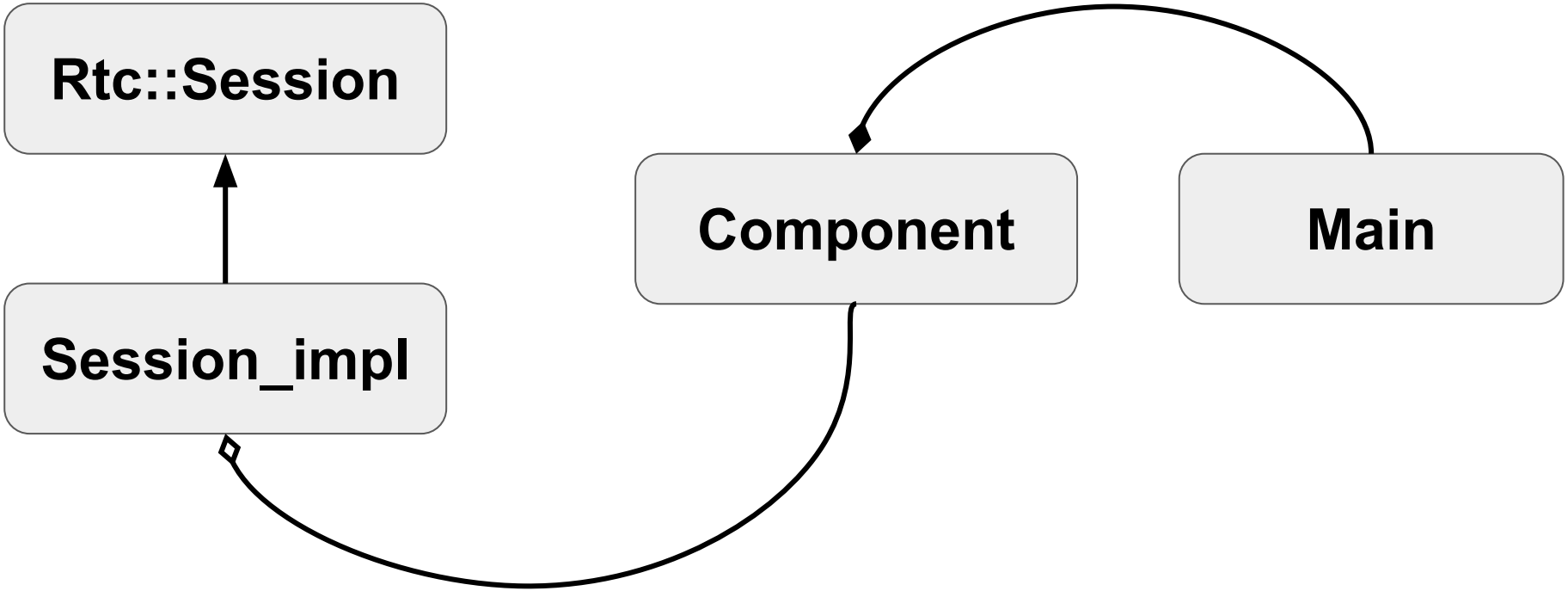
Структура компонента mock_rtc



Структура компонента mock_rtc



Структура компонента mock_rtc



Структура компонента mock_rtc

Rtc::Session

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

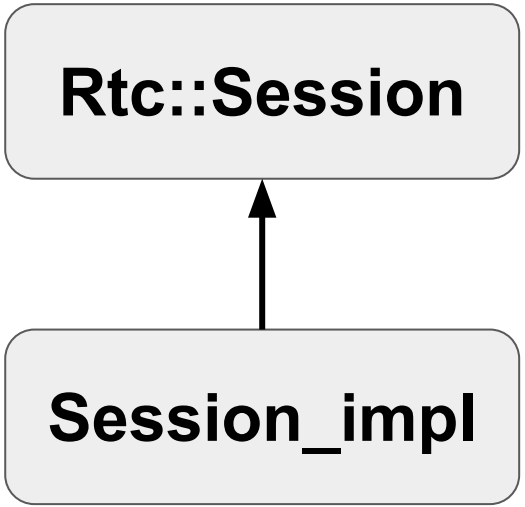
```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2      static const char *service_name() { return "Rtc"; }
 3      enum { CAP_QUOTA = 2 };
 4
 5      virtual void set_sigh(
 6          Signal_context_capability sigh) = 0;
 7      virtual Timestamp current_time() = 0;
 8      GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                 Signal_context_capability);
10      GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11      GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

```
// genode/repos/os/include/rtc_session/rtc_session.h
 1  struct Rtc::Session : Genode::Session {
 2     static const char *service_name() { return "Rtc"; }
 3     enum { CAP_QUOTA = 2 };
 4
 5     virtual void set_sigh(
 6        Signal_context_capability sigh) = 0;
 7     virtual Timestamp current_time() = 0;
 8     GENODE_RPC(Rpc_set_sigh, void, set_sigh,
 9                Signal_context_capability);
10     GENODE_RPC(Rpc_current_time, Timestamp, current_time);
11     GENODE_RPC_INTERFACE(Rpc_set_sigh, Rpc_current_time);
12  };
```

Структура компонента mock_rtc



```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
 1  struct Session_impl : Rpc_object<Rtc::Session> {
 2      Timestamp ts;
 3      Signal_context_capability sig_cap { };
 4
 5      Session_impl(Timestamp const &ts) : ts(ts) { }
 6      Timestamp current_time() override { return ts; }
 7      void set_sigh(Signal_context_capability sigh) override {
 8          sig_cap = sigh; }
 9
10      void set_timestamp(Timestamp const &ts);
11      void notify_client();
12  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
 1  struct Session_impl : Rpc_object<Rtc::Session> {
 2      Timestamp ts;
 3      Signal_context_capability sig_cap { };
 4
 5      Session_impl(Timestamp const &ts) : ts(ts) { }
 6      Timestamp current_time() override { return ts; }
 7      void set_sigh(Signal_context_capability sigh) override {
 8          sig_cap = sigh; }
 9
10      void set_timestamp(Timestamp const &ts);
11      void notify_client();
12  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
 1  struct Session_impl : Rpc_object<Rtc::Session> {
 2      Timestamp ts;
 3      Signal_context_capability sig_cap { };
 4
 5      Session_impl(Timestamp const &ts) : ts(ts) { }
 6      Timestamp current_time() override { return ts; }
 7      void set_sigh(Signal_context_capability sigh) override {
 8          sig_cap = sigh; }
 9
10      void set_timestamp(Timestamp const &ts);
11      void notify_client();
12  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
 1  struct Session_impl : Rpc_object<Rtc::Session> {
 2      Timestamp ts;
 3      Signal_context_capability sig_cap { };
 4
 5      Session_impl(Timestamp const &ts) : ts(ts) { }
 6      Timestamp current_time() override { return ts; }
 7      void set_sigh(Signal_context_capability sigh) override {
 8          sig_cap = sigh; }
 9
10      void set_timestamp(Timestamp const &ts);
11      void notify_client();
12  };
```

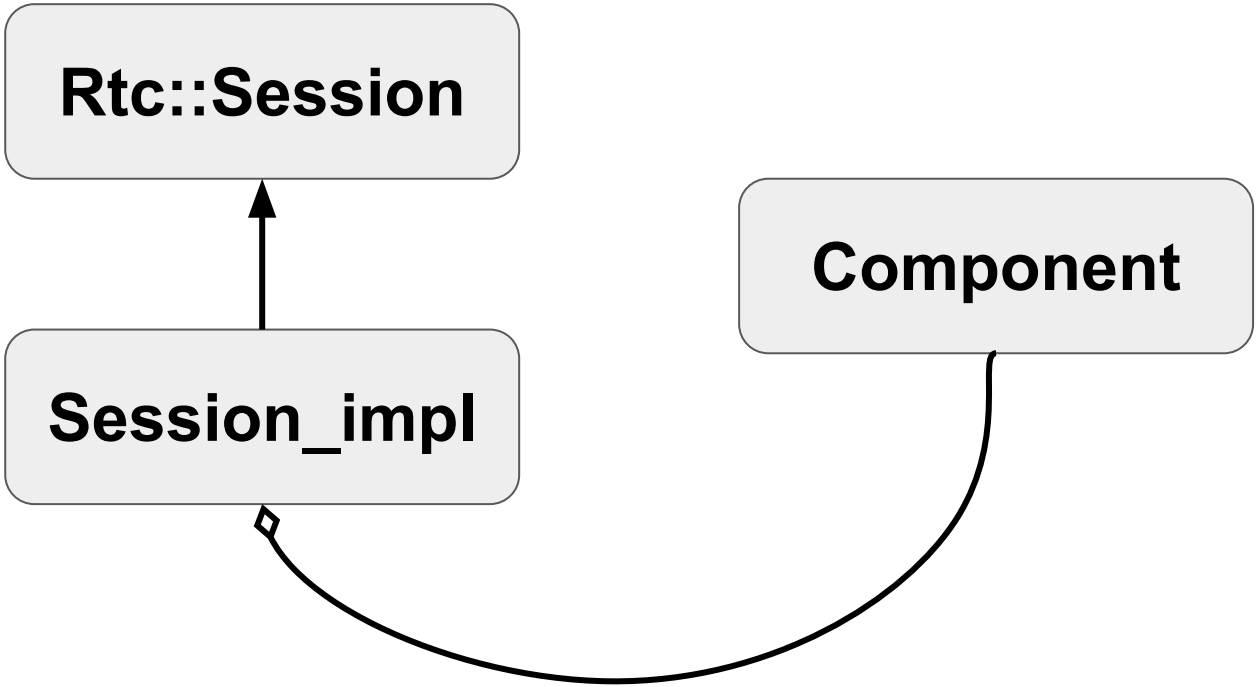
```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
 1  struct Session_impl : Rpc_object<Rtc::Session> {
 2      Timestamp ts;
 3      Signal_context_capability sig_cap { };
 4
 5      Session_impl(Timestamp const &ts) : ts(ts) { }
 6      Timestamp current_time() override { return ts; }
 7      void set_sigh(Signal_context_capability sigh) override {
 8          sig_cap = sigh; }
 9
10      void set_timestamp(Timestamp const &ts);
11      void notify_client();
12  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
13  void Session_impl::set_timestamp(Timestamp const &ts) {
14      this->ts = ts;
15      notify_client();
16  }
17
18  void Session_impl::notify_client() {
19      if (sig_cap.valid()) {
20          Signal_transmitter(sig_cap).submit();
21      }
22  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
13  void Session_impl::set_timestamp(Timestamp const &ts) {
14      this->ts = ts;
15      notify_client();
16  }
17
18  void Session_impl::notify_client() {
19      if (sig_cap.valid()) {
20          Signal_transmitter(sig_cap).submit();
21      }
22  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
13  void Session_impl::set_timestamp(Timestamp const &ts) {
14      this->ts = ts;
15      notify_client();
16  }
17
18  void Session_impl::notify_client() {
19      if (sig_cap.valid()) {
20          Signal_transmitter(sig_cap).submit();
21      }
22  }
```

Структура компонента mock_rtc



```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
23  using Root_component = Genode::Root_component<Session_impl>;
24  struct Component : Root_component {
25      Timestamp ts { };
26      Registery<Registered<Session_impl>> sessions { };
27
28      Session_impl *_create_session(const char *) override;
29
30      void set_timestamp(Timestamp const &ts);
31
32      Component(Env &env, Allocator &md_alloc) :
33          Root_component(&env.ep().rpc_ep(), &md_alloc) { }
34  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
35  Session_impl *Component::_create_session(const char *) {
36      return new (md_alloc())
37          Registered<Session_impl>(sessions, ts);
38  }
39
40  void Component::set_timestamp(Timestamp const &ts) {
41      this->ts = ts;
42      sessions.for_each([this] (Session_impl &session) {
43          session.set_timestamp(this->ts); });
44  }
```

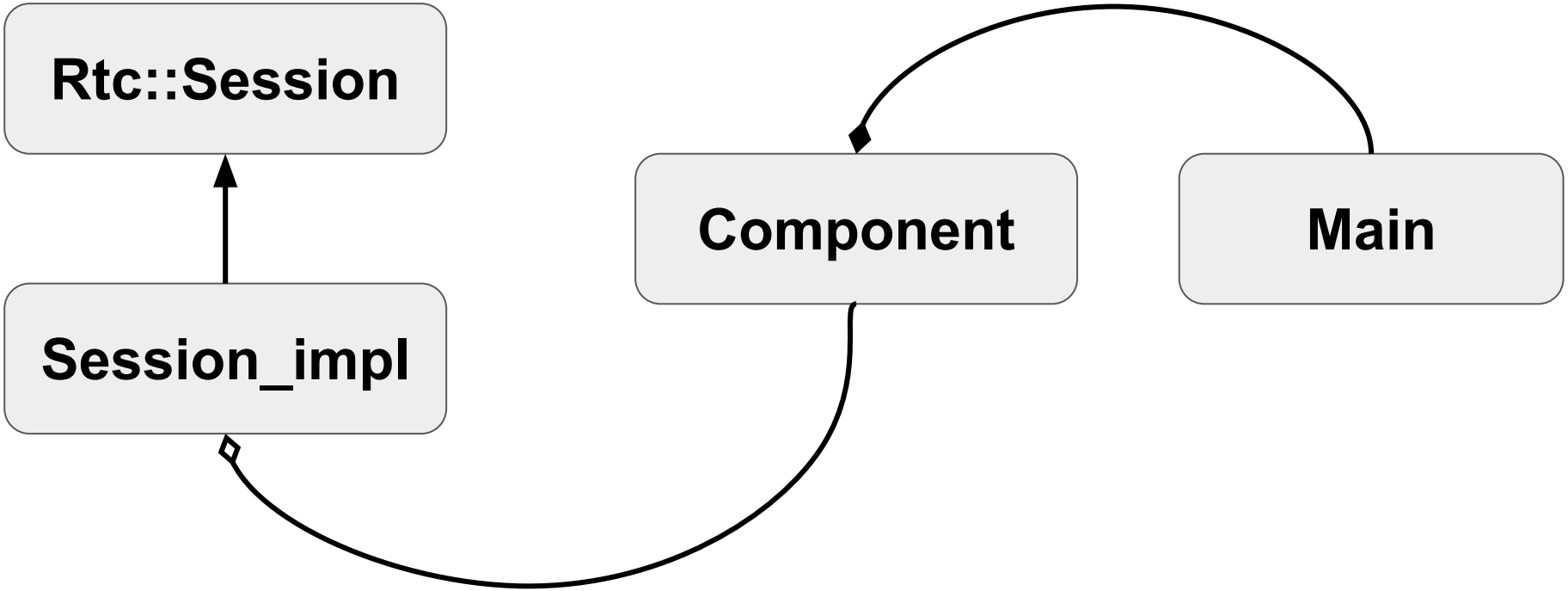
```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
35  Session_impl *Component::_create_session(const char *) {
36      return new (md_alloc())
37          Registered<Session_impl>(sessions, ts);
38  }
39
40  void Component::set_timestamp(Timestamp const &ts) {
41      this->ts = ts;
42      sessions.for_each([this] (Session_impl &session) {
43          session.set_timestamp(this->ts); });
44  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
35  Session_impl *Component::_create_session(const char *) {
36      return new (md_alloc())
37          Registered<Session_impl>(sessions, ts);
38  }
39
40  void Component::set_timestamp(Timestamp const &ts) {
41      this->ts = ts;
42      sessions.for_each([this] (Session_impl &session) {
43          session.set_timestamp(this->ts); });
44  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
35  Session_impl *Component::_create_session(const char *) {
36      return new (md_alloc())
37          Registered<Session_impl>(sessions, ts);
38  }
39
40  void Component::set_timestamp(Timestamp const &ts) {
41      this->ts = ts;
42      sessions.for_each([this] (Session_impl &session) {
43          session.set_timestamp(this->ts); });
44  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
35  Session_impl *Component::_create_session(const char *) {
36      return new (md_alloc())
37          Registered<Session_impl>(sessions, ts);
38  }
39
40  void Component::set_timestamp(Timestamp const &ts) {
41      this->ts = ts;
42      sessions.for_each([this] (Session_impl &session) {
43          session.set_timestamp(this->ts); });
44  }
```

Структура компонента mock_rtc



```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
45  struct Main {
46      Sliced_heap              sliced_heap;
47      Attached_rom_dataspace   config_rom;
48      Component                root;
49      Signal_handler<Main>     timer_handler;
50      Timer::Connection        timer;
51
52      void on_timeout();
53      Main(Env &env);
54  };
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
55  Main::Main(Env &env) :
56      sliced_heap   { env.ram(), env.rm() },
57      config_rom    { env, "config" },
58      root          { env, sliced_heap },
59      timer_handler { env.ep(), *this, &Main::on_timeout },
60      timer         { env }
61  {
62      root.set_timestamp(parse_xml(config_rom.xml()));
63      timer.sigh(timer_handler);
64      timer.trigger_periodic(1'000'000);
65      env.parent().announce(env.ep().manage(root));
66  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
67  void Main::on_timeout() {
68      root.set_timeout(root.ts + 10'000'000);
69  }
70
71  void Component::construct(Genode::Env &env) {
72      static Rtc::Main main(env);
73  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
67  void Main::on_timeout() {
68      root.set_timeout(root.ts + 10'000'000);
69  }
70
71  void Component::construct(Genode::Env &env) {
72      static Rtc::Main main(env);
73  }
```

```
// genode/repos/cpp-russia-2021/src/test/mock_rtc/main.cc
67  void Main::on_timeout() {
68      root.set_timeout(root.ts + 10'000'000);
69  }
70
71  void Component::construct(Genode::Env &env) {
72      static Rtc::Main main(env);
73  }
```

Genode: конфигурация init

```
15      <start name="mock_rtc" caps="100">
16          <resource name="RAM" quantum="4M"/>
17          <provides> <service name="Rtc"/> </provides>
18          <config year="2021" month="11" day="16" />
19          <route>
20              <service name="Timer"> <child name="timer"/>
21              </service>
22              <any-service> <parent/> </any-service>
23          </route>
24      </start>
```

Genode: конфигурация init

```
15      <start name="mock_rtc" caps="100">
16          <resource name="RAM" quantum="4M"/>
17          <provides> <service name="Rtc"/> </provides>
18          <config year="2021" month="11" day="16" />
19          <route>
20              <service name="Timer"> <child name="timer"/>
21              </service>
22              <any-service> <parent/> </any-service>
23          </route>
24      </start>
```

Genode: конфигурация init

```
15      <start name="mock_rtc" caps="100">
16          <resource name="RAM" quantum="4M"/>
17          <provides> <service name="Rtc"/> </provides>
18          <config year="2021" month="11" day="16" />
19          <route>
20              <service name="Timer"> <child name="timer"/>
21              </service>
22              <any-service> <parent/> </any-service>
23          </route>
24      </start>
```

Genode: конфигурация init

```
15      <start name="mock_rtc" caps="100">
16          <resource name="RAM" quantum="4M"/>
17          <provides> <service name="Rtc"/> </provides>
18          <config year="2021" month="11" day="16"/>
19          <route>
20              <service name="Timer"> <child name="timer"/>
21              </service>
22              <any-service> <parent/> </any-service>
23          </route>
24      </start>
```

Genode: конфигурация init

```
15      <start name="mock_rtc" caps="100">
16          <resource name="RAM" quantum="4M"/>
17          <provides> <service name="Rtc"/> </provides>
18          <config year="2021" month="11" day="16" />
19          <route>
20              <service name="Timer"> <child name="timer"/>
21              </service>
22              <any-service> <parent/> </any-service>
23          </route>
24      </start>
```

log_time.run

```
25  <start name="log_time" caps="200">
26      <resource name="RAM" quantum="4M"/>
27      <config>
28          <libc stdin="/dev/null" stdout="/dev/log"
29                stderr="/dev/log" rtc="/dev/rtc"/>
30          <vfs>
31              <dir name="dev"><log/><null/><rtc/></dir>
32          </vfs>
33      </config>
```

log_time.run

```
25  <start name="log_time" caps="200">
26      <resource name="RAM" quantum="4M"/>
27      <config>
28          <libc stdin="/dev/null" stdout="/dev/log"
29                stderr="/dev/log" rtc="/dev/rtc"/>
30          <vfs>
31              <dir name="dev"><log/><null/><rtc/></dir>
32          </vfs>
33      </config>
```

log_time.run

```
34      <route>
35          <service name="Rtc">
36              <child name="mock_rtc"/>
37          </service>
38          <service name="Timer">
39              <child name="timer"/>
40          </service>
41          <any-service> <parent/> </any-service>
42      </route>
43  </start>
```

log_time.run

```
34      <route>
35          <service name="Rtc">
36              <child name="mock_rtc"/>
37          </service>
38          <service name="Timer">
39              <child name="timer"/>
40          </service>
41          <any-service> <parent/> </any-service>
42      </route>
43  </start>
```

rtc_timer.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    mock_rtc posix.lib.so stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

rtc_timer.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    mock_rtc posix.lib.so stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

rtc_timer.run

```
1  create_boot_directory
2  build { core init app/log_time test/mock_rtc timer }
3
4  install_config { $init_config }
5
6  build_boot_image {
7    core init ld.lib.so libc.lib.so libm.lib.so log_time
8    mock_rtc posix.lib.so stdcxx.lib.so timer vfs.lib.so }
9
10 run_genode_until forever
```

Запускаем log_time.run

```
$ make KERNEL=linux BOARD=linux -C build/x86_64 run/log_time
...
[init -> log_time] 2021-11-17 00:00:00
[init -> log_time] 2021-11-17 00:00:10
[init -> log_time] 2021-11-17 00:00:20
[init -> log_time] 2021-11-17 00:00:30
...
```

Итоги

Итоги

Как запускать libc приложение

Итоги

Как запускать libc приложение

Как создать сервис

Итоги

Как запускать libc приложение

Как создать сервис

Как связать приложение и сервис

Итоги

Как запускать libc приложение

Как создать сервис

Как связать приложение и сервис

Как использовать сессии для тестирования

Спасибо за внимание!

https://twitter.com/sermp
https://t.me/sermp
https://github.com/sergey-platonov

https://genode.org/

https://github.com/genodelabs

http://genodians.org/

users@lists.genode.org

https://www.reddit.com/r/genode/

https://genode.org/download/sculpt

log_time.run

```
13          <vfs>
14              <dir name="dev"><log/><null/><rtc/></dir>
15          </vfs>
16      </config>
17      <route>
18          <service name="Rtc">
19              <child name="rtc_timer"/>
20          </service>
21          <any-service> <parent/> </any-service>
22      </route>
```

log_time.run

```
24      <route>
25          <service name="Timer">
26              <child name="timer"/>
27          </service>
28          <any-service> <parent/> </any-service>
29      </route>
30  </start>
31  </config>
```