VirtualBox:
ssh -p 8000 seso@127.0.0.1   (username: seso | password: oses)

VMWare:
sudo dhclient
ip a
ssh seso@dirección_IP   (username: seso | password: oses)
==========

SESO/
- kernel/
- tools/
  - cmake-tool/
- build/
- projects/
  - musllibc/
  - utils_libs/
  - seL4_libs/
  - seso/   #código generado
    - src/

```
mkdir SESO
cd SESO
git clone --branch 10.1.1 https://github.com/seL4/seL4.git kernel
git clone --branch 10.1.x-compatible https://github.com/seL4/seL4_tools.git tools
mkdir projects
git clone --branch 10.1.x-compatible https://github.com/seL4/seL4_libs.git projects/sel4_libs
git clone --branch 10.1.x-compatible https://github.com/seL4/musllibc.git projects/musllibc
git clone --branch 10.1.x-compatible https://github.com/seL4/util_libs.git projects/util_libs
ln -s tools/cmake-tool/init-build.sh init-build.sh          #en SESO
nano CMakeLists.txt   #en SESO
```

```
cmake_minimum_required(VERSION 3.7.2)
if (${PLATFORM} IN_LIST KernelX86Sel4Arch_all_strings)
    set(KernelArch x86 CACHE STRING "" FORCE)
    set(KernelX86Sel4Arch ${PLATFORM} CACHE STRING "" FORCE)
endif()
include(tools/cmake-tool/default-CMakeLists.txt)
if(SIMULATION)
    ApplyCommonSimulationSettings("x86")
else()
    if(KernelArchX86)
        set(KernelIOMMU ON CACHE BOOL "" FORCE)
    endif()
endif()
# We must build the debug kernel because the tutorials rely on
# seL4_DebugPutChar
# and they don't initialize a platsupport driver.
ApplyCommonReleaseVerificationSettings(FALSE FALSE)
GenerateSimulateScript()
```

```
mkdir build
mkdir projects/seso
mkdir projects/seso/src
```

cd projects/seso/src
nano main.c

```
#include <stdio.h>
int main(void)
{
        printf("Bienvenido al sistema operativo SESO\n");
        return 0;
}
```

cd ~/SESO/projects/seso
nano CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7.2)
project(SESO C) # create a new C project called 'Hello'
# add files to our project. Paths are relative to this file.
add_executable(SESO src/main.c)
# we need to link against the standard C lib for printf
target_link_libraries(SESO sel4muslcsys muslc)
# Set this image as the rootserver
DeclareRootserver(SESO)
```

cd build
../init-build.sh -DPLATFORM=x86_64 -DSIMULATION=TRUE
ninja
./simulate # Para salir de la simulación se tiene que pulsar <ctrl>+<a> y luego <x>

==========

nano main.c

```
#include <stdio.h>
#include <sel4/sel4.h>
#include <sel4platsupport/bootinfo.h>

const seL4_BootInfo *boot_info;

static void print_bootinfo(const seL4_BootInfo* info) {
        int i;

        /* General info */
        printf("Info Page: %p\n", info);
        printf("IPC Buffer: %p\n", info->ipcBuffer);
        printf("Node ID: %d (of %d)\n", info->nodeID, info->numNodes);
        printf("IOPT levels: %d\n", info->numIOPTLevels);
        printf("Init cnode size bits: %d\n", info->initThreadCNodeSizeBits);

        /* Cap details */
        printf("\nCap details:\n");
        printf("Type Start End\n");
        printf("Empty 0x%08x 0x%08x\n", info->empty.start, info->empty.end);
        printf("Shared frames 0x%08x 0x%08x\n", info->sharedFrames.start, info-
        >sharedFrames.end);
        printf("User image frames 0x%08x 0x%08x\n", info->userImageFrames.start, info-
        >userImageFrames.end);
        printf("User image PTs 0x%08x 0x%08x\n", info->userImagePaging.start, info-
        >userImagePaging.end);
        printf("Untypeds 0x%08x 0x%08x\n", info->untyped.start, info->untyped.end);

        /* Untyped details */
        printf("-----------------------------------------------------\n");
```

```
        printf("\nUntyped details:\n");
        printf("Untyped Slot Paddr Bits Device\n");
        for (i = 0; i < info->untyped.end-info->untyped.start; i++) {
            if (!(info->untypedList[i].isDevice))
                printf(" %3d 0x%08x 0x%08x %2d %d\n", i, info->untyped.start +
            i, info->untypedList[i].paddr, info->untypedList[i].sizeBits, info-
            >untypedList[i].isDevice);
        }
        printf("-------------------------------------------------------\n");
    }
    int main(void) {
        printf(">>>\n>>> Bienvenido al sistema operativo SESO\n>>>\n");
        boot_info = platsupport_get_bootinfo();
        print_bootinfo(boot_info);
        return 0;
    }
```

==========

```
// Inicializa el sistema de memoria según el tipo de alineación.
int init_memory_system(seL4_Uint8 aligment);

// La función allocate() asignará una región de memoria del tamaño 2^sizeBits bytes
// alineada. Si la región solicitada no está disponible, la función imprimirá un mensaje de
// error y cancelará la ejecución.
seL4_Word allocate(seL4_Uint8 sizeBits);

//La función release() liberará la región de la dirección indicada.
int release(seL4_Word paddr);
```