

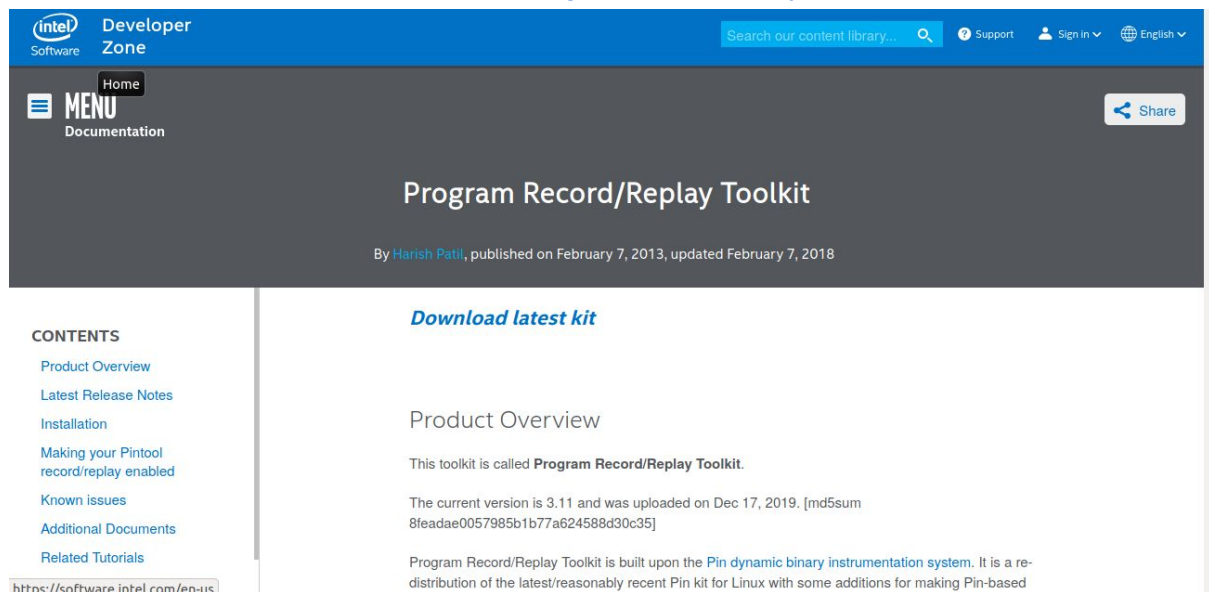
# SINUCA-tracer

Este documento descreve os passos necessários para a utilização do SINUCA-tracer dentro do OrCS para gerar traços de execução de aplicações ou executar qualquer aplicação diretamente com o OrCS.

## Download do Pin + PinPlay

A versão mais recente dessas ferramentas pode ser baixada no seguinte endereço:

<https://software.intel.com/en-us/articles/program-recordreplay-toolkit>



O arquivo baixado deve ser colocado dentro do diretório **trace\_generator** do OrCS.

## Configurações

Após o download e posicionamento do PinPoints, o arquivo **install\_tracer.sh** presente no diretório raiz do OrCS deve ser executado. Este arquivo faz toda a instalação do PinPoints dentro do OrCS.

# Compilando o Sinuca Tracer

Para compilar o gerador de traços, navegue até a pasta:

```
>> trace_generator/extras/pinplay/sinuca_tracer
```

E digite os seguintes comandos:

```
>> make clean
```

```
>> make
```

O resultado esperado da compilação sem erros se parecerá com isto:

```
*****
Moving instruction_type_count.so to /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/
mv obj-intel64/instruction_type_count.so /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/

g++ -shared -Wl,--hash-style=sysv /home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt/crtbeginS.o -Wl,-Bsymbolic -Wl,--version-script=/home/war/Mestrado/transferencia/OrCS/trace_generator/source/include/pin/pintool.ver -fabi-version=2 -o obj-intel64/library_call.so obj-intel64/library_call.o /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib/intel64/libpinplay.a /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib-ext/intel64/libbz2.a /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib-ext/intel64/libzlib.a /home/war/Mestrado/transferencia/OrCS/trace_generator/source/tools/InstLib/obj-intel64/controller.a -L/home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt -L/home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/lib -L/home/war/Mestrado/transferencia/OrCS/trace_generator/extras/xed-intel64/lib -Lpin -Lxed /home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt/crtendS.o -lpin3dwarf -ldl -dynamic -nostdlib -lstdport -dynamic -ln-dynamic -lc-dynamic -lunwind-dynamic

*****
Moving library_call.so to /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/
mv obj-intel64/library_call.so /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/

g++ -shared -Wl,--hash-style=sysv /home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt/crtbeginS.o -Wl,-Bsymbolic -Wl,--version-script=/home/war/Mestrado/transferencia/OrCS/trace_generator/source/include/pin/pintool.ver -fabi-version=2 -o obj-intel64/profile.so obj-intel64/profile.o /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib/intel64/libpinplay.a /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib-ext/intel64/libbz2.a /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/lib-ext/intel64/libzlib.a /home/war/Mestrado/transferencia/OrCS/trace_generator/source/tools/InstLib/obj-intel64/controller.a -L/home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt -L/home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/lib -L/home/war/Mestrado/transferencia/OrCS/trace_generator/extras/xed-intel64/lib -Lpin -Lxed /home/war/Mestrado/transferencia/OrCS/trace_generator/intel64/runtime/pincrt/crtendS.o -lpin3dwarf -ldl -dynamic -nostdlib -lstdport -dynamic -ln-dynamic -lc-dynamic -lunwind-dynamic

*****
Moving profile.so to /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/
mv obj-intel64/profile.so /home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/bin/intel64/

make[1]: Leaving directory '/home/war/Mestrado/transferencia/OrCS/trace_generator/extras/pinplay/sinuca_tracer'
```

## Criando traços

Após a compilação, traços podem ser gerados através do seguinte comando, executado a partir da pasta **trace\_generator/extras/pinplay/sinuca\_tracer**:

```
>> ../../pin -t ../bin/intel64/sinuca_tracer.so -trace x86 -- <<Caminho até o executável>>
```

### Exemplo:

```
>> ../../pin -t ../bin/intel64/sinuca_tracer.so -trace x86 -- /bin/ls
```

Esse comando gera uma saída parecida com a seguinte:

```
gcc threads = 1
Inserted Output File Name = output_trace.out
Real Static File = output_trace.out.tid0.stat.out.gz => READY !
Real Dynamic File = output_trace.out.tid0.dyn.out.gz => READY !
Real Memory File = output_trace.out.tid0.mem.out.gz => READY !
Loading /bin/ls, Image id = 1
Loading /lib64/ld-linux-x86-64.so.2, Image id = 2
Loading [vdso], Image id = 3
Loading /lib/x86_64-linux-gnu/libselinux.so.1, Image id = 4
Loading /lib/x86_64-linux-gnu/libc.so.6, Image id = 5
Loading /lib/x86_64-linux-gnu/libpcre.so.3, Image id = 6
Loading /lib/x86_64-linux-gnu/libdl.so.2, Image id = 7
Loading /lib/x86_64-linux-gnu/libpthread.so.0, Image id = 8
branch_profiler.cpp      intrinsics_extension.hpp  obj-ia32                output_trace.out.tid0.mem.out.gz  profile.cpp
defines.hpp              library_call.cpp           obj-intel64             output_trace.out.tid0.stat.out.gz sinuca_tracer.cpp
enumerations.hpp         makefile                   opcode_package.hpp       pinplay-debugger-shell.cpp
instruction_type_count.cpp makefile.rules              opcodes.hpp             pinplay-debugger-shell.H
intrinsic_extensions.cpp memory_request_client.hpp  output_trace.out.tid0.dyn.out.gz pinplay-driver.cpp
```

E cria 3 arquivos .gz que são os traços criados que podem ser utilizados como entrada para simulações com a atual versão do OrCS.

## Executando diretamente

Um programa pode ser executado diretamente sobre o OrCS. Para isso, o OrCS deve ser compilado com os comandos:

```
>> make clean
```

```
>> make
```

(O sinuca tracer também deve ter sido compilado)

Em seguida, o seguinte comando pode ser utilizado na pasta raiz do OrCS (a mesma do make que compilou o OrCS):

```
>> trace_generator/pin -t trace_generator/extras/pinplay/bin/intel64/direct_tracer.so -c  
<arquivo de configuração do OrCS> -f <resultados do OrCS> -- <programa a ser  
executado> <argumentos do programa>
```

Por exemplo, com o seguinte comando podemos simular o ls sendo executado em uma arquitetura Sandy Bridge com o OrCS:

```
>> trace_generator/pin -t trace_generator/extras/pinplay/bin/intel64/direct_tracer.so -c  
config/sandy_bridge/sandy_bridge.cfg -f result.temp -- /bin/l
```

Notas:

- Caso o arquivo de configuração não seja informado, a configuração padrão é a do Sandy Bridge.
- Apenas aplicações single thread podem ser executadas desta maneira.
- A saída padrão do programa simulado é o stdout (seu terminal).
- Caso o arquivo de resultados não seja definido (flag -f), o arquivo **results.res** será utilizado como argumento -f para o OrCS.

Versões utilizadas:

<b>Pin + PinPlay</b>	3.11
<b>GCC</b>	8.4.0
<b>Sistema operacional</b>	Ubuntu 18.04.1 LTS