Portable Executables

Linus Hagemann and Tom Wollnik

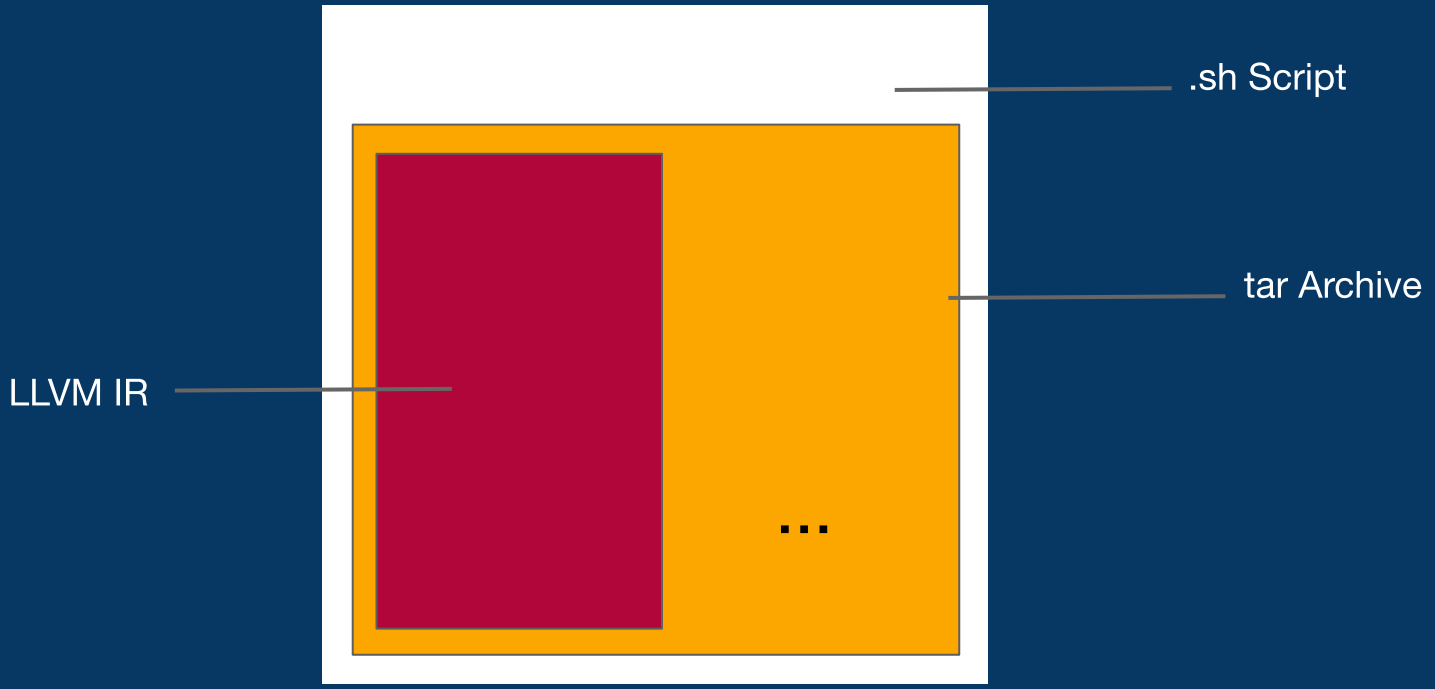
Betriebssysteme II

August 4th 2020

Demo Time

Motivation

PEX Format



LLVM IR

```c
#include <stdio.h>

int main() {
    printf("Hello World!");
    return 0;
}
```

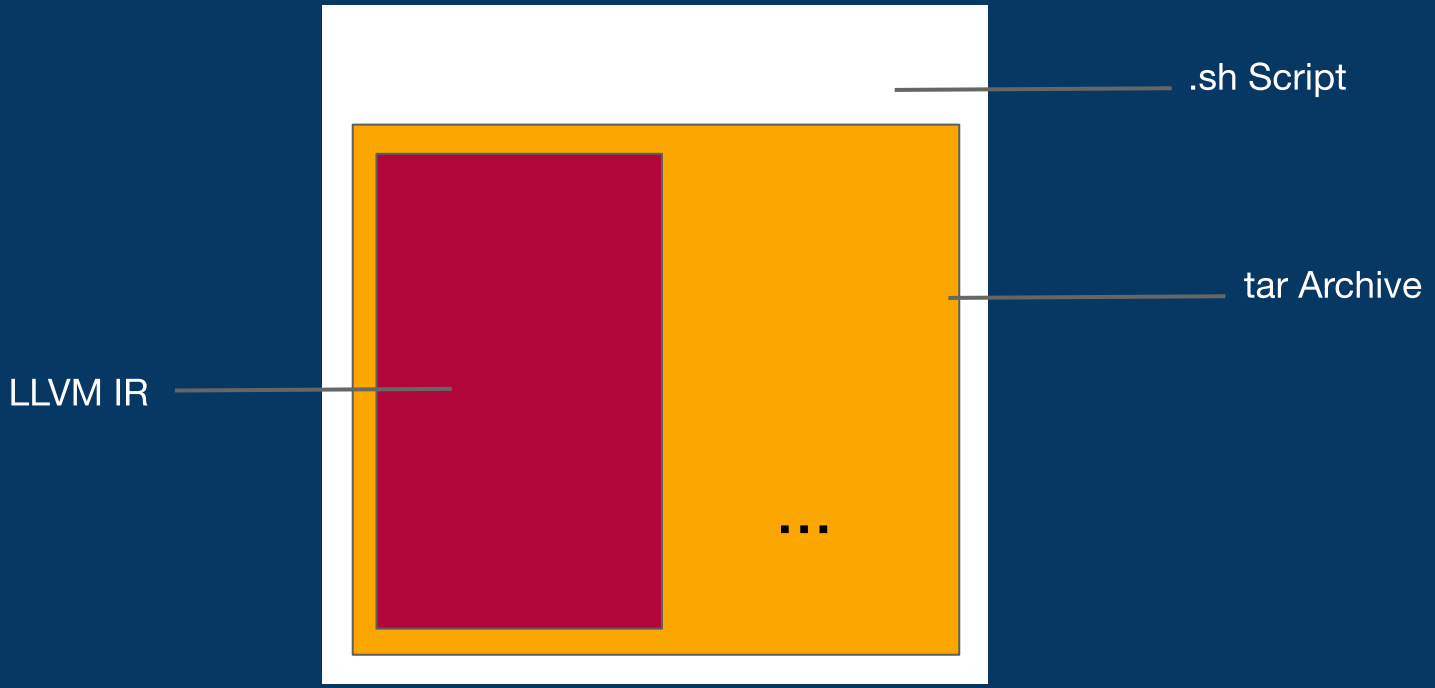
```
; ModuleID = 'hello.c'
source_filename = "hello.c"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@.str = private unnamed_addr constant [13 x i8] c"Hello World!\00", align 1

; Function Attrs: noinline nounwind optnone uwtable
define i32 @main() #0 {
  %1 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %2 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([13 x i8], [13 x i8]*
@.str, i32 0, i32 0))
  ret i32 0
}

declare i32 @printf(i8*, ...) #1

attributes #0 = {[...] "target-cpu"="x86-64"
"target-features"="+fxsr,+mmx,+sse,+sse2,+x87" [...] }
[...]
```

```
#include <stdio.h>

int main() {
   printf("Hello World!");
   return 0;
}
```

```
; ModuleID = 'hello.c'
source_filename = "hello.c"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@.str = private unnamed_addr constant [13 x i8] c"Hello World!\00", align 1

; Function Attrs: noinline nounwind optnone uwtable
define i32 @main() #0 {
  %1 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %2 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([13 x i8], [13 x i8]*
@.str, i32 0, i32 0))
  ret i32 0
}

declare i32 @printf(i8*, ...) #1

attributes #0 = {[...] "target-cpu"="x86-64"
"target-features"="+fxsr,+mmx,+sse,+sse2,+x87" [...] }
[...]
```

```
#include <stdio.h>

int main() {
   printf("Hello World!");
   return 0;
}
```

PEX Format



Limitations

Compilers and build processes are complex.

LLVM IR is not completely architecture agnostic.

PEX has performance and memory overhead.

Future Work

Future Work

Recursive PEX?

Integration with OS to improve performance

Future Work

Recursive PEX?

Integration with OS to improve performance

Real IR Portability?

Integration with Compiler Toolchain