预备

1. Fork ArceOS的工程，clone到本地。工程链接如下

   ```
   git@github.com:arceos-org/arceos.git
   ```

2. 在main分支下，创建并切换到新的分支week1，执行

   ```
   git checkout -b week1
   ```

   后面的实验都在该分支下进行。

练习1

支持彩色打印println!。以apps/helloworld为测试应用。

要求：不能在helloworld程序本身改，要在下面的库或更深层次的组件修改。

预期输出：执行`make run ARCH=riscv64`，输出的效果：

```
arch = riscv64
platform = riscv64-qemu-virt
target = riscv64gc-unknown-none-elf
smp = 1
build_mode = release
log_level = warn

Hello, world!
```

练习2(附加题)

支持HashMap数据类型。以apps/memtest为测试应用。

首先修改apps/memtest/src/main.rs，把BTreeMap替换为HashMap，如下：

```
 use rand::{rngs::SmallRng, RngCore, SeedableRng};
-use std::collections::BTreeMap;
+use std::collections::HashMap;
 use std::vec::Vec;

 fn test_vec(rng: &mut impl RngCore) {
@@ -22,9 +22,9 @@ fn test_vec(rng: &mut impl RngCore) {
     println!("test_vec() OK!");
 }

-fn test_btree_map(rng: &mut impl RngCore) {
+fn test_hashmap_map(rng: &mut impl RngCore) {
     const N: usize = 50_000;
-    let mut m = BTreeMap::new();
+    let mut m = HashMap::new();
     for _ in 0..N {
         let value = rng.next_u32();
         let key = format!("key_{value}");
@@ -35,7 +35,7 @@ fn test_btree_map(rng: &mut impl RngCore) {
```

```
             assert_eq!(k.parse::<u32>().unwrap(), *v);
         }
     }
-    println!("test_btree_map() OK!");
+    println!("test_hashmap_map() OK!");
 }

 #[cfg_attr(feature = "axstd", no_mangle)]
@@ -44,7 +44,7 @@ fn main() {

     let mut rng = SmallRng::seed_from_u64(0xdead_beef);
     test_vec(&mut rng);
-    test_btree_map(&mut rng);
+    test_hashmap_map(&mut rng);

     println!("Memory tests run OK!");
 }
```

然后，尝试编译运行，`make A=apps/memtest ARCH=riscv64 run`，此时会报错，因为我们目前不支持HashMap类型。

要求：在ulib/axstd中支持HashMap类型

预期输出：执行`make A=apps/memtest ARCH=riscv64 run`

```
arch = riscv64
platform = riscv64-qemu-virt
target = riscv64gc-unknown-none-elf
smp = 1
build_mode = release
log_level = warn

Running memory tests...
test_vec() OK!
test_hashmap_map() OK!
Memory tests run OK!
```

提示：

1. 参考官方rust标准库中的HashMap实现，把涉及的代码拷过来，做一下修改。只需要满足memtest的测试需要即可。
2. 注意：官方std与ArceOS的axstd的区别。官方rust标准库主要是基于Linux/Windows这些内核，为应用提供的用户库。官方std的支持后端是libc+syscall；而ArceOS是单特权级，没有syscall一说，axstd直接通过一系列function-call调用底层的功能。
3. HashMap之所以没有像其他collections类型一样放到alloc库中实现，主要是因为它需要随机数的支持，而随机数的产生机制是平台相关的。大家做实验可以简单点，用一个软实现的随机数函数来产生。比如

   ```
   use spinlock::SpinNoIrq;
   use crate::time;

   static PARK_MILLER_LEHMER_SEED: SpinNoIrq<u32> = SpinNoIrq::new(0);
   const RAND_MAX: u64 = 2_147_483_647;
   ```

```
pub fn random() -> u128 {
    let mut seed = PARK_MILLER_LEHMER_SEED.lock();
    if *seed == 0 {
        *seed = time::current_ticks() as u32;
    }

    let mut ret: u128 = 0;
    for _ in 0..4 {
        *seed = ((u64::from(*seed) * 48271) % RAND_MAX) as u32;
        ret = (ret << 32) | (*seed as u128);
    }
    ret
}
```

能够完成练习2的同学，请把你完成工作的github工程链接发到下面的邮箱

```
sun_ye@masscIouds.com
```