Fuchsia 翻译小组文档翻译测试（2022）

请**翻译**和**校对**的报名者完成本英译中测试。

本测试中提供的材料基于 Fuchsia 最新英文文档内容，无标准答案、不公布打分结果。

您只需要从本测试提供的 3 个部分（part）中**选择 1 个部分**完成翻译，不必完成所有部分。为此，您可以先 通览一遍各部分的引导语，选择更有把握、更感兴趣的内容进行翻译。

您填写的答案将由人工审核，作为您报名的辅助信息。因此您不必过于紧张，只需按照您的理解进行翻译即可。

注意：

1. 请尽可能采用直译方式，减少意译的使用。
2. 请尽可能用简洁直白的语言准确翻译内容，不必追求辞藻华丽、意境优美。
3. 您可以使用机器翻译工具作为翻译的**辅助**手段，亦可使用互联网工具查找有利于您进行翻译工作的任何资料。
4. 请您自觉地独立完成本翻译测试，不要求助他人。

请在填写完成后将文档保存，重命名为您填写的个人称呼，并通过电子邮件发送至 whyto@fuchsia-china.com，邮件主题请填写“翻译测试PDF提交”。

*** 必填**

请填写您在报名表中填写的称呼 *

此信息用于将该测试与报名者对应

请填写您在报名表中填写的电子邮箱 *

此信息用于将该测试与报名者对应和接收回执邮件

NOTE: You do NOT need to necessarily read the reference information if you can already simply translate the passages.

PART I

You will translate some passages about the C++ programming language taken from one document. Here is the abstract of the document for your reference.

From development/languages/c-cpp/compile_time_object_collections.md

This document covers active discussion about building compile-time collections of objects in C++. The following use cases are examples of where compile-time collections are useful:

- StringRef - A type that supports building a compile-time collection of string labels with associated unique numeric ids for tracing purposes.
- LockClass - A type that supports building a compile-time collection of state objects for runtime lock validation purposes.

The following sections discuss common and unique requirements of each use case, the current challenges with the implementations, and proposed solutions.

1. Please translate the following passage into Simplified Chinese.

StringRef is a type that implements the concept of string references. A string reference is a mapping from a numeric id to a character string. Using the mapping makes more economical use of the trace buffer: an (id, string) pair is emitted once in a tracing session and then subsequent events may refer to the string by id instead of including the full character sequence inline.

2. Please translate the following passage into Simplified Chinese.

LockClass is a type that captures information about a lock that is common to all instances of the lock (e.g. its containing type if it is a struct/class member, the type of the underlying lock primitive, flags describing its behavior). The LockClass type is used by the runtime lock validator to determine which ordering rules apply to each lock and to locate the per-lock-class tracking structure used to record ordering observations.

PART II
You will translate some passages about some USB driver concepts and processes in Fuchsia. Here is a piece of overall information for your reference.

From development/drivers/driver_guides/usb/concepts/overview.md

Zircon provides a full featured USB subsystem enabling the development of USB host and peripheral devices. Low, full, high, and super-speed devices are supported as well as various standard autonegotiation mechanisms.

3. Please translate the following passage into Simplified Chinese.

From development/drivers/driver_guides/usb/concepts/overview.md

The first step in a USB request's lifecycle is allocation. USB requests contain data from all of the drivers in the request stack in a single allocation. Each driver that is upstream of a USB device driver should provide a GetRequestSize method -- which returns the size it needs to contain its local request context. When a USB device driver allocates a request, it should invoke this method to determine the size of the parent's request context.

4. Please translate the following passage into Simplified Chinese.

From development/drivers/driver_guides/usb/concepts/overview.md

In general, USB device drivers encode transfer requests into a usb_request_t structure. These request structs generally have an asynchronous callback associated with them to be executed upon transfer completion. For the most part, the USB stack functions by the higher order device drivers publishing requests to a queue of outstanding requests. As these requests are serviced, their respective callbacks are invoked notifying the upper layers that the request is complete.

PART III
You will translate some passages about some kernel concepts in Fuchsia taken from one document. Here is a piece of overall information for your reference.

From concepts/kernel/README.md

Zircon is the core platform that powers Fuchsia. Zircon is composed of a kernel (source in /zircon/kernel) as well as a small set of userspace services, drivers, and libraries (source in /zircon/system/) necessary for the system to boot, talk to hardware, load userspace processes and run them, etc. Fuchsia builds a much larger OS on top of this foundation.

The Zircon Kernel provides syscalls to manage processes, threads, virtual memory, inter-process communication, waiting on object state changes, and locking (via futexes).

5. Please translate the following passage into Simplified Chinese.

From concepts/kernel/handles.md

Handles are kernel constructs that allow user-mode programs to reference a kernel object. A handle can be thought of as a session or connection to a particular kernel object.
It is often the case that multiple processes concurrently access the same object via different handles. However, a single handle can only be either bound to a single process or be bound to the kernel.

6. Please translate the following passage into Simplified Chinese.

From concepts/kernel/concepts.md

Threads represent threads of execution (CPU registers, stack, etc) within an address space that is owned by the Process in which they exist. Processes are owned by Jobs, which define various resource limitations. Jobs are owned by parent Jobs, all the way up to the Root Job, which was created by the kernel at boot and passed to "userboot", the first userspace Process to begin execution.