4.3 Types of Prerequisites

There are actually two different types of prerequisites understood by GNU make: normal prerequisites such as described in the previous section, and `order-only` prerequisites. A normal prerequisite makes two statements: first, it imposes an order in which recipes will be invoked: the recipes for all prerequisites of a target will be completed before the recipe for the target is run. Second, it imposes a dependency relationship: if any prerequisite is newer than the target, then the target is considered out-of-date and must be rebuilt.

GNU make 实际上有两种不同类型的先决条件：正常的先决条件，如上一节中描述的，以及仅订购的先决条件。 一个正常的先决条件有两个陈述：首先，它规定了调用配方的顺序：目标的所有先决条件的配方将在目标的配方运行之前完成。 其次，它强加了一种依赖关系：如果任何先决条件比目标更新，那么目标被认为是过时的，必须重新构建。

Normally, this is exactly what you want: if a target’s prerequisite is updated, then the target should also be updated.

通常，这正是您想要的：如果更新了目标的先决条件，那么也应该更新目标。

Occasionally, however, you have a situation where you want to impose a specific ordering on the rules to be invoked without forcing the target to be updated if one of those rules is executed. In that case, you want to define order-only prerequisites. Order-only prerequisites can be specified by placing a pipe symbol (|) in the prerequisites list: any prerequisites to the left of the pipe symbol are normal; any prerequisites to the right are order-only:

```
targets : normal-prerequisites | order-only-prerequisites
```

但是，有时您会遇到这样的情况，即您希望对要调用的规则施加特定的顺序，而不是在执行这些规则之一时强制更新目标。 在这种情况下，您想要定义仅订单的先决条件。 可以通过在先决条件列表中放置管道符号 (|) 来指定仅订单先决条件：管道符号左侧的任何先决条件都是正常的； 右侧的任何先决条件都是仅订购的：

```
targets : normal-prerequisites | order-only-prerequisites
```

The normal prerequisites section may of course be empty. Also, you may still declare multiple lines of prerequisites for the same target: they are appended appropriately (normal prerequisites are appended to the list of normal prerequisites; order-only prerequisites are appended to the list of order-only prerequisites). Note that if you declare the same file to be both a normal and an order-only prerequisite, the normal prerequisite takes precedence (since they have a strict superset of the behavior of an order-only prerequisite).

正常的先决条件部分当然可能是空的。 此外，您仍然可以为同一目标声明多行先决条件：它们被适当地附加（正常先决条件附加到正常先决条件列表中；仅订单先决条件附加到仅订单先决条件列表中）。 请注意，如果您将同一文件声明为普通和仅订单前提条件，则普通前提条件优先（因为它们具有仅订单前提条件的行为的严格超集）。

Consider an example where your targets are to be placed in a separate directory, and that directory might not exist before make is run. In this situation, you want the directory to be created before any targets are placed into it but, because the timestamps on directories change whenever a file is added, removed, or renamed, we certainly don't want to rebuild all the targets whenever the directory's timestamp changes. One way to manage this is with order-only prerequisites: make the directory an order-only prerequisite on all the targets:

```
OBJDIR := objdir
OBJS := $(addprefix $(OBJDIR)/,foo.o bar.o baz.o)

$(OBJDIR)/%.o : %.c
        $(COMPILE.c) $(OUTPUT_OPTION) $<

all: $(OBJS)

$(OBJS): | $(OBJDIR)

$(OBJDIR):
        mkdir $(OBJDIR)
```

考虑一个示例，您的目标将被放置在一个单独的目录中，并且在运行 make 之前该目录可能不存在。 在这种情况下，您希望在将任何目标放入其中之前创建目录，但是，因为每当添加、删除或重命名文件时目录上的时间戳都会更改，所以我们当然不希望在任何时候重建所有目标当目录的时间戳更改。 管理此问题的一种方法是仅订购先决条件：使目录成为所有目标的仅订购先决条件：

```
OBJDIR := objdir
OBJS := $(addprefix $(OBJDIR)/,foo.o bar.o baz.o)

$(OBJDIR)/%.o : %.c
        $(COMPILE.c) $(OUTPUT_OPTION) $<

all: $(OBJS)

$(OBJS): | $(OBJDIR)
```

```
$(OBJDIR):
        mkdir $(OBJDIR)
```

Now the rule to create the objdir directory will be run, if needed, before any ‘.o’ is built, but no ‘.o’ will be built because the objdir directory timestamp changed.

现在，如果需要，将在构建任何 “.o” 之前运行创建 objdir 目录的规则，但不会构建 “.o”，因为 objdir 目录时间戳已更改。
