一、准备镜像

本文采用最小化安装方式，同时会阐述OpenRC和SystemD两种init环境安装区别。所以，镜像采用的是最小化minimal镜像+Stage3(OpenRC/SystemD)，整个过程需要有公网访问能力，因此Stage3镜像不用下载到本地，直接在live系统中wget即可。镜像可在国内 **清华Gentoo镜像站下载**。

1.OpenRC需要准备的镜像

访问**OpenRC下载地址**，下载minimal镜像：

File Name ↓	File Size ↓	Date ↓
Parent directory/	-	-
admincd-amd64-20230108T161708Z.iso	622.1 MiB	2023-01-09 19:51
admincd-amd64-20230108T161708Z.iso.CONTENTS.gz	9.9 KiB	2023-01-09 19:51
admincd-amd64-20230108T161708Z.iso.DIGESTS	1.2 KiB	2023-01-15 08:01
admincd-amd64-20230108T161708Z.iso.asc	488 B	2023-01-09 20:01
admincd-amd64-20230108T161708Z.iso.sha256	652 B	2023-01-15 08:01
install-amd64-minimal-20230108T161708Z.iso	432.2 MiB	2023-01-09 02:35
install-amd64-minimal-20230108T161708Z.iso.CONTENTS.gz	9.9 KiB	2023-01-09 02:35
install-amd64-minimal-20230108T161708Z.iso.DIGESTS	1.3 KiB	2023-01-15 08:01
install-amd64-minimal-20230108T161708Z.iso.asc	488 B	2023-01-09 03:01
install-amd64-minimal-20230108T161708Z.iso.sha256	660 B	2023-01-15 08:01
livegui-amd64-20230108T161708Z.iso	3.6 GiB	2023-01-09 08:03
livegui-amd64-20230108T161708Z.iso.CONTENTS.gz	10.0 KiB	2023-01-09 08:03
livegui-amd64-20230108T161708Z.iso.DIGESTS	1.2 KiB	2023-01-15 08:01

以及同目录下的OpenRC Stage3镜像:

stage3-amd64-openrc-20230108T161708Z.tar.xz	228.7 MiB	2023-01-09 01:26
stage3-amd64-openrc-20230108T161708Z.tar.xz.CONTENTS.gz	636.6 KiB	2023-01-09 01:26
stage3-amd64-openrc-20230108T161708Z.tar.xz.DIGESTS	1.3 KiB	2023-01-15 08:01
stage3-amd64-openrc-20230108T161708Z.tar.xz.asc	488 B	2023-01-09 01:41
stage3-amd64-openrc-20230108T161708Z.tar.xz.sha256	661 B	2023-01-15 08:01

> 可选步骤，这里只要清楚需要哪几个镜像即可，Stage3不用下载到本地。

2.SystemD需要准备的镜像

访问**SystemD下载地址**，SystemD作为init系统时，Stage3镜像则为：

stage3-amd64-systemd-20230108T161708Z.tar.xz	248.6 MiB	2023-01-09 03:02
stage3-amd64-systemd-20230108T161708Z.tar.xz.CONTENTS.gz	696.3 KiB	2023-01-09 03:02
stage3-amd64-systemd-20230108T161708Z.tar.xz.DIGESTS	1.3 KiB	2023-01-15 08:01
stage3-amd64-systemd-20230108T161708Z.tar.xz.asc	488 B	2023-01-09 03:21
stage3-amd64-systemd-20230108T161708Z.tar.xz.sha256	662 B	2023-01-15 08:01

minimal镜像还是没有变：

install-amd64-minimal-20230108T161708Z.iso	432.2 MiB	2023-01-09 02:35
install-amd64-minimal-20230108T161708Z.iso.CONTENTS.gz	9.9 KiB	2023-01-09 02:35
install-amd64-minimal-20230108T161708Z.iso.DIGESTS	1.3 KiB	2023-01-15 08:01
install-amd64-minimal-20230108T161708Z.iso.asc	488 B	2023-01-09 03:01
install-amd64-minimal-20230108T161708Z.iso.sha256	660 B	2023-01-15 08:01

3.OpenRC和SystemD的优势对比

1）OpenRC优势

- 具有更低的内存占用，对于低端服务器和资源受限系统来说，OpenRC是一个不错的选择；
- 易于管理和维护，因为它是一个脚本系统，可以轻松通过修改脚本来配置系统；
- 配置文件容易理解，OpenRC使用普通的Shell脚本，它的配置文件相对更容易理解；

2）SystemD优势

- 更快的启动速度，可以更快的启动和停止服务；
- 更多的功能，支持多用户启动，支持多种服务状态的检测和管理，支持热插拔硬件，支持远程管理等；
- 更高的可靠性，SystemD可以更好地处理系统故障，并可以更好地分析和诊断系统问题；

4.选用OpenRC还是SystemD

OpenRC是Gentoo的原生init系统，核心部份处理依赖管理和init脚本分析，通过扫描运行级别，建造依赖图，接着启动需要的服务脚本来工作，启动方式为 `rc-service <service> <action>` 或者 `/etc/init.d/<service> <action>`：

```
 ⊙ 21:10:26  ~  rc-service sshd status
 * status: started
 ⊙ 21:11:00  ~  /etc/init.d/sshd status
 * status: started
 ⊙ 21:11:03  ~  inxi -v4|grep -Po ".*\KInit.*$"
Init: SysVinit rc: OpenRC
 ⊙ 21:11:10  ~  _
```

SystemD，由Fedora 15引入，相对OpenRC较新的init系统，也是当今最流行的init系统，集成了各类工具以便更好的管理系统，目前在各大Linux发行版基本都能见到，绝大部分默认都是以SystemD作为init系统，启动速度比OpenRC快很多，启动方式为 `systemctl [Option] [Unit]`：

```
 ⊙ 21:13:45  ~  inxi -v4|grep -Po ".*\KInit.*$"
Init: systemd
 ⊙ 21:14:06  ~  systemctl status sshd
● sshd.service - OpenSSH server daemon
     Loaded: loaded (/lib/systemd/system/sshd.service; enabled; preset: disabled)
     Active: active (running) since Sat 2023-01-14 23:58:46 CST; 21h ago
   Main PID: 641 (sshd)
      Tasks: 1 (limit: 9501)
     Memory: 4.6M
        CPU: 229ms
     CGroup: /system.slice/sshd.service
             └─641 "sshd: /usr/sbin/sshd -D -e [listener] 0 of 10-100 startups"
```

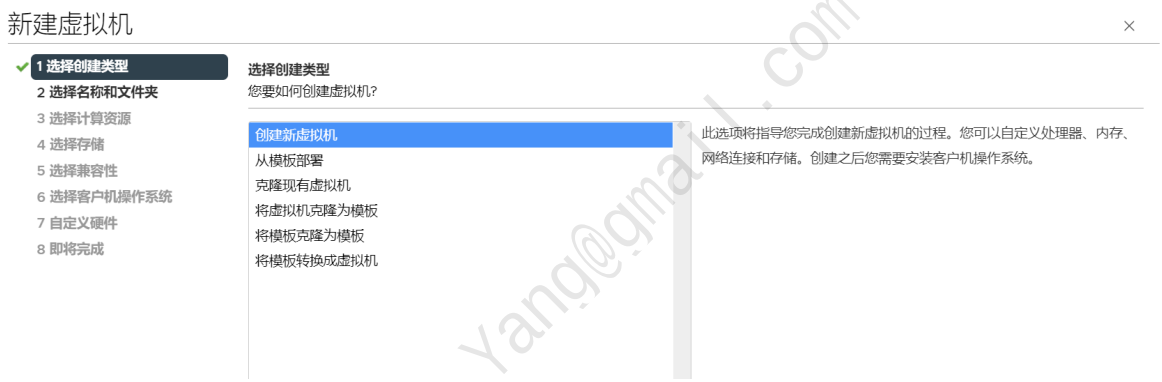
OpenRC是Gentoo的默认Init系统，Gentoo官网整个安装步骤也是围绕此展开讨论，SystemD是Gentoo添加上去的支持，因此安装SystemD需要一些额外的步骤，但个人建议是选用SystemD集大成者，除非你清晰知道OpenRC更适合你的需求场景。

二、创建ESXi虚拟机

将minimal镜像上传到ESXi后，接下来可开始创建虚拟机，已有相关经验的可跳过这一步。

> 注：这里是用vCenter作为操作平台，界面展示上和ESXi有所区别，但实际就是操纵ESXi。

1.选择创建类型



2.选择名称和文件夹

这里理解为将Gentoo安装到哪一台ESXi上面，如果你直接在ESXi操作则没有此步骤。

3.选择计算资源

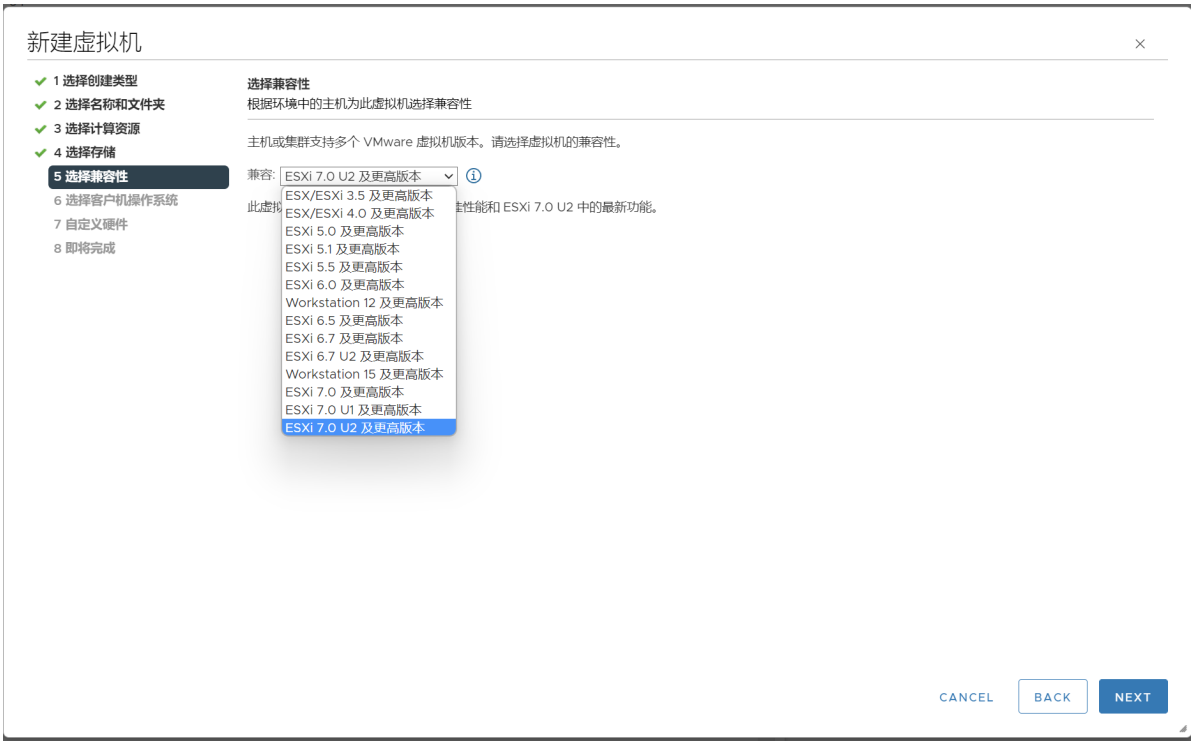
选择集群机器，101则为ESXi地址：

4.选择存储

这里将Gentoo安装到16TB的硬盘里：

5.选择兼容性

选默认最高版本即可，这里用的是ESXi 7.0：



6.选择客户机操作系统

操作系统版本：Linux

客户机操作系统版本：其他5.x或更高版本的Linux(64位)

7.自定义硬件

根据主机性能酌情分配，并挂载好minimal镜像。Gentoo安装软件默认都会把源码拉下来编译，性能可以尽量给高点，提升安装效率，编译内核速度也会更快。

虚拟机选项-->引导选项，这里选择BIOS：

三、进入虚拟机并配置网络

1.进入虚拟机

打开虚拟机后，默认第一个LiveCD选项进入系统：

2.配置网络

如果你在【自定义硬件】步骤选用的网络适配器能上公网，那么此步骤可忽略。

使用 `ifconfig` 或 `ip addr` 查看网卡名：

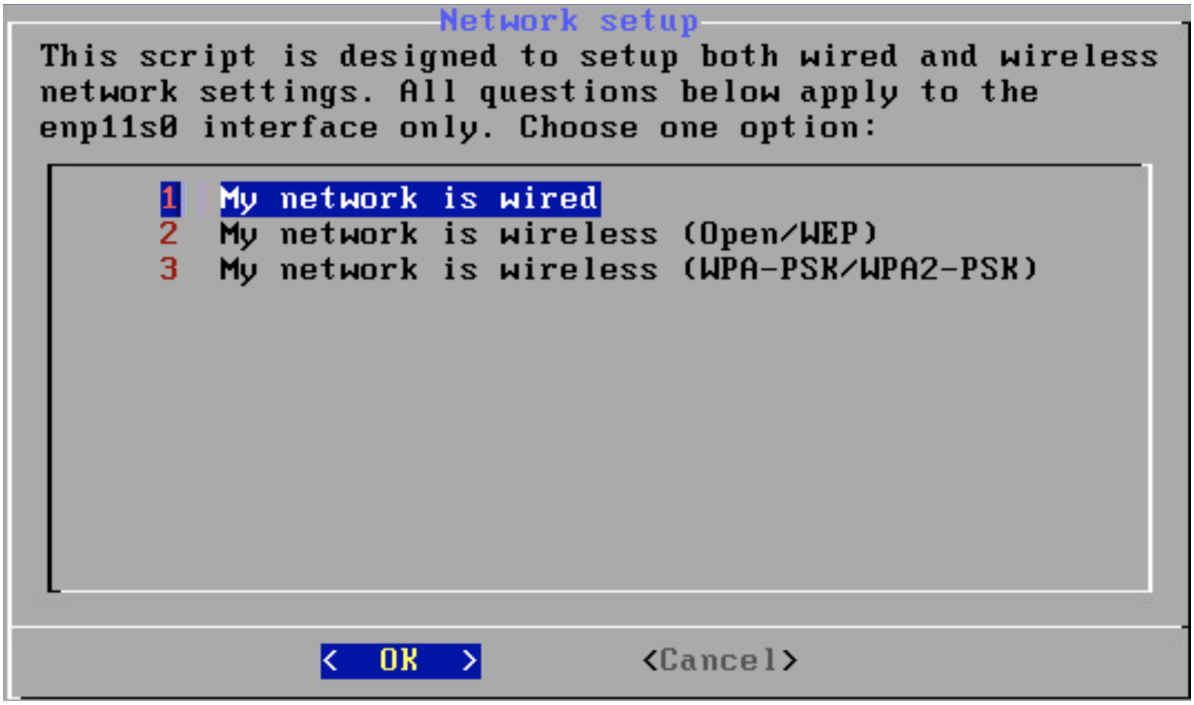
```
(none) ~ # ifconfig
enp11s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.221  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6332:dbe1:325d:124e  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:81:41:27  txqueuelen 1000  (Ethernet)
        RX packets 1664  bytes 116961 (114.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 51  bytes 3394 (3.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

可以看到Live系统默认已经从DHCP地址池中获取到了IP，一般这一步都没有太大问题。

测试公网连通性及DNS解析也没有问题：

```
(none) ~ # ping baidu.com
PING baidu.com (39.156.66.10) 56(84) bytes of data.
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=1 ttl=50 time=42.5 ms
From openwrt.linux-code.com (192.168.1.1): icmp_seq=2 Redirect Host(New nexthop:
 ros.linux-code.com (192.168.1.11))
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=2 ttl=50 time=44.6 ms
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=3 ttl=50 time=42.5 ms
```

如果没网，可以考虑使用 `net-setup` 来配置，比如上面网卡是enp11s0，那么命令则为： `net-setup enp11s0` ，输出为下面的dialog界面，按照界面提示选择网络环境即可：



配置网络部分不做赘述，可参考 **官方文档**。

3.开启sshd

设置一个密码，并开启sshd来远程登录，这样直接方便复制操纵一些文本：

```
(none) ~ # passwd
New password:
Retype new password:
passwd: password updated successfully
(none) ~ # rc-service sshd start
 * Starting sshd ...
(none) ~ #
```

通过终端ssh进入到live系统后的效果：

```
(none) ~ # ifconfig
enp11s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.221  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6332:dbe1:325d:124e  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:81:41:27  txqueuelen 1000  (Ethernet)
        RX packets 3800  bytes 275320 (268.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 208  bytes 23079 (22.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

四、配置分区

1.分区方案

以100G磁盘为例，分区方案可以如下作为参考：

分区	大小	描述
/dev/sda1	256M	/boot分区
/dev/sda2	2G	交换（swap）分区，一般为内存的两倍，内存足够大可以考虑适量给。
/dev/sda3	剩余部分	根分区

2.使用fdisk分区

选择parted也没啥问题，大于2T的硬盘用不了fdisk。

使用如下命令开始分区硬盘：

```
fdisk /dev/sda
```

如果你用的是SSD，那么硬盘名称形如 `/dev/nvme0n1` 。

1）创建/boot分区

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-209715199, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-209715199, default 209715199):
+256M

Created a new partition 1 of type 'Linux' and of size 256 MiB.
```

2）创建swap分区

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
```

```
Select (default p): p
Partition number (2-4, default 2):
First sector (526336-209715199, default 526336):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (526336-209715199, default 209715199):
+2G

Created a new partition 2 of type 'Linux' and of size 2 GiB.

Command (m for help): t
Partition number (1,2, default 2):
Hex code or alias (type L to list all): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'.
```

3）创建根分区

```
Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (3,4, default 3):
First sector (4720640-209715199, default 4720640):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (4720640-209715199, default 209715199):

Created a new partition 3 of type 'Linux' and of size 97.7 GiB.

Command (m for help): p
Disk /dev/sda: 100 GiB, 107374182400 bytes, 209715200 sectors
Disk model: Virtual disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xaf2138c5

Device     Boot   Start       End   Sectors  Size Id Type
/dev/sda1          2048    526335    524288  256M 83 Linux
/dev/sda2        526336   4720639   4194304    2G 82 Linux swap / Solaris
/dev/sda3       4720640 209715199 204994560 97.7G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

```
(none) ~ # fdisk /dev/sda

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xaf2138c5.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-209715199, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-209715199, default 209715199): +256M

Created a new partition 1 of type 'Linux' and of size 256 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (526336-209715199, default 526336):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (526336-209715199, default 209715199): +2G

Created a new partition 2 of type 'Linux' and of size 2 GiB.

Command (m for help): t
Partition number (1,2, default 2):
Hex code or alias (type L to list all): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'.

Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (3,4, default 3):
First sector (4720640-209715199, default 4720640):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (4720640-209715199, default 209715199):

Created a new partition 3 of type 'Linux' and of size 97.7 GiB.

Command (m for help): p
Disk /dev/sda: 100 GiB, 107374182400 bytes, 209715200 sectors
Disk model: Virtual disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xaf2138c5

Device     Boot   Start       End   Sectors  Size Id Type
/dev/sda1          2048    526335    524288  256M 83 Linux
/dev/sda2        526336   4720639   4194304    2G 82 Linux swap / Solaris
/dev/sda3       4720640 209715199 204994560 97.7G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

(none) ~ # _
```

3.创建文件系统

比如，在示例分区结构中，有使用FAT32的引导分区（/dev/sda1）和使用ext4的根分区（/dev/sda3），则使用下面的命令：

```
mkfs.vfat -F 32 /dev/sda1
mkfs.ext4 /dev/sda3
```

激活并初始化swap分区：

```
mkswap /dev/sda2
swapon /dev/sda2
```

```
(none) ~ # mkfs.vfat -F 32 /dev/sda1
mkfs.fat 4.2 (2021-01-31)
(none) ~ # mkfs.ext4 /dev/sda3
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 25624320 4k blocks and 6406144 inodes
Filesystem UUID: 7a05ddf2-b16c-4b4f-a70b-50288aededa8
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

(none) ~ # mkswap /dev/sda2
Setting up swapspace version 1, size = 2 GiB (2147479552 bytes)
no label, UUID=e483307b-a5ce-44d2-94d6-d88b0584b5e0
(none) ~ # swapon /dev/sda2
(none) ~ #
```

4.挂载root分区

使用mount命令将根分区挂载到目录下面：

```
mount /dev/sda3 /mnt/gentoo
```

```
(none) ~ # mount /dev/sda3 /mnt/gentoo
(none) ~ # df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
none           tmpfs      16G  784K   16G   1% /run
udev           devtmpfs   10M     0   10M   0% /dev
tmpfs          tmpfs      16G     0   16G   0% /dev/shm
tmpfs          tmpfs      16G   65M   16G   1% /
/dev/sr0       iso9660   432M  432M     0 100% /mnt/cdrom
/dev/loop0     squashfs  390M  390M     0 100% /mnt/livecd
cgroup_root    tmpfs      10M     0   10M   0% /sys/fs/cgroup
/dev/sda3      ext4       96G   24K   91G   1% /mnt/gentoo
(none) ~ # _
```

五、安装Stage包

1.设置日期和时间

建议和 `UTC` 时区保持一致，注意不是 `UTC+8` ，Gentoo网络基础服务使用了安全证书，如果系统时间差的离谱，可能无法下载安装文件，后面步骤会设置为正确的时区。

可以使用内置的ntpd命令和ntp服务器同步：

```
ntpd -q -g
```

或者你也可以手动设置，比如设置时间为2023年1月16日18:00，则为：

```
date 011618002023
```

2.下载Stage归档文件

1）OpenRC

进入Gentoo的根挂载点：

```
cd /mnt/gentoo
```

下载OpenRC Stage3包文件:

```
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-openrc-20230108T161708Z.tar.xz
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-openrc-20230108T161708Z.tar.xz.CONTENTS.gz
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-openrc-20230108T161708Z.tar.xz.DIGESTS
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-openrc-20230108T161708Z.tar.xz.asc
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-openrc-20230108T161708Z.tar.xz.sha256
```

验证文件下载是否完整，避免文件下载不完全导致后面步骤白做，比如使用SHA512校验：

```
openssl dgst -r -sha512 stage3-amd64-openrc-20230108T161708Z.tar.xz
cat stage3-amd64-openrc-20230108T161708Z.tar.xz.DIGESTS
```

```
(none) /mnt/gentoo # openssl dgst -r -sha512 stage3-amd64-openrc-20230108T161708Z.tar.xz
c2307bcdd559f26c69589a294c779dca42c8282e9963cba54855757bf7d245d28df3e49ceb8dc2b6b29c7febe7c2a26ddb00b85afb3fa2bd356111dc6e73d5fb *s
tage3-amd64-openrc-20230108T161708Z.tar.xz
(none) /mnt/gentoo # cat stage3-amd64-openrc-20230108T161708Z.tar.xz.DIGESTS
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

# BLAKE2B HASH
f696f486ea8718721a60ce3e77df4c06c9e98519915c68ebd8780ecd9cb683763ac8ce6b01f48fd70ee7ee33fa7159905a92e644b0cec6eb8e4f70e9d5861fcd  s
tage3-amd64-openrc-20230108T161708Z.tar.xz
# SHA512 HASH
c2307bcdd559f26c69589a294c779dca42c8282e9963cba54855757bf7d245d28df3e49ceb8dc2b6b29c7febe7c2a26ddb00b85afb3fa2bd356111dc6e73d5fb  s
tage3-amd64-openrc-20230108T161708Z.tar.xz
# BLAKE2B HASH
a7d105e583920a7d9298c7ef928858e2a9ef899e6a9e56dec964a522383a67bfa7b280c3ce0c23223ad55b0b23f8f4324885d61fe2110d63e8e2b555fc299365  s
tage3-amd64-openrc-20230108T161708Z.tar.xz.CONTENTS.gz
# SHA512 HASH
d563513dbd9128536960ba8de6929e8f496cf2062b58fc1f879f4579229d87864a706a9471d1e59f0c2e54ab0c98ff59945c16e7f42b227cd35ade739013a9c4  s
tage3-amd64-openrc-20230108T161708Z.tar.xz.CONTENTS.gz
-----BEGIN PGP SIGNATURE-----

iQEzBAEBCAAdFiEEU05CCatJ7uHBnZYWLERpXbn2BD0FAmPEk8IACgkQLERpXbn2
BD1zXAf/Uqv9daIZsNSdgL6WDX+gsqSIrp/Z5Nynjo3nA5ssckP2Feo+0DZnYUCn
NzK0p191WLwG9XA6bKB4zNlVh0C0pgdtgVhMGwWKBTRiiuLauDew9tvXehnE6m6i
6DsbnGp7KjPNO+125BZK+4XqiLmHPxFVklhxJgsPPBfmikrH6CLv4h2jQSa/Yb16
BcsL99Wi/frdLAda/YbRajSN5Clsy0TaFlV/nAJH9zTsFNx2RFCN82dfrlqKqf6k
iFfdWEC++JbGu1Zl0YBRVYu65wwLWPGQ6Buxin/I5nloYqHfUzHSkgW4rINGktC0
INqI9GA8tvSJr8YOFAh0ubmO8WrpEA==
=vlZL
-----END PGP SIGNATURE-----
(none) /mnt/gentoo # _
```

确保完全一致即可。

2）SystemD

进入Gentoo的根挂载点：

```
cd /mnt/gentoo
```

下载SystemD Stage3包文件：

```
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-systemd-20230108T161708Z.tar.xz
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-systemd-20230108T161708Z.tar.xz.CONTENTS.gz
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-systemd-20230108T161708Z.tar.xz.DIGESTS
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-systemd-20230108T161708Z.tar.xz.asc
wget
https://mirrors.tuna.tsinghua.edu.cn/gentoo/releases/amd64/autobuilds/20230108T161708Z
/stage3-amd64-systemd-20230108T161708Z.tar.xz.sha256
```

使用SHA256校验文件是否完整：

```
openssl dgst -r -sha256 stage3-amd64-systemd-20230108T161708Z.tar.xz
cat stage3-amd64-systemd-20230108T161708Z.tar.xz.sha256
```

```
(none) /mnt/gentoo # ls
lost+found                                          stage3-amd64-systemd-20230108T161708Z.tar.xz.DIGESTS
stage3-amd64-systemd-20230108T161708Z.tar.xz        stage3-amd64-systemd-20230108T161708Z.tar.xz.asc
stage3-amd64-systemd-20230108T161708Z.tar.xz.CONTENTS.gz  stage3-amd64-systemd-20230108T161708Z.tar.xz.sha256
(none) /mnt/gentoo # openssl dgst -r -sha256 stage3-amd64-systemd-20230108T161708Z.tar.xz
3f6914e17ee61c32bf5c140d06b11217f2d8a91c90bde78ed3f61ca3041bcf45 *stage3-amd64-systemd-20230108T161708Z.tar.xz
(none) /mnt/gentoo # cat stage3-amd64-systemd-20230108T161708Z.tar.xz.sha256
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

# SHA256 HASH
3f6914e17ee61c32bf5c140d06b11217f2d8a91c90bde78ed3f61ca3041bcf45 stage3-amd64-systemd-20230108T161708Z.tar.xz
-----BEGIN PGP SIGNATURE-----

iQEzBAEBCAAdFiEEU05CCatJ7uHBnZYWLERpXbn2BD0FAmPEk8IACgkQLERpXbn2
BDlKsgf+I/ODN8Pnwa7aWhrMf4cnly8D+U8O6xTJrkVeVj/ziIqwlSYXvh66qgSr
WSuUt10iYzCOLyH550vgbON0Sv4iTvYqc1UrTPLVt4pVg22LW0sPv5CWohDLuReu
U1QldIOAytvp54XxKUBaUzfjMv5dhzAh1UHFo1+TRSSlhVlQkICnjNqPdIGN3Jul
apc5RxJI9cubpXMMMbEoBl3JVsXsg0CbwBmMxxxYQbaEIdgklO5BCnyQlN1hkCMK
MqRxhnwphT/SLAg4iJs1k8E6HPUabPvWpbNvmppzuEmDnqBPfCpX4tT98sevRFm5
9tH+IB1xJ79ER8Qu3RCYKl9TaR/Jaw==
=khA0
-----END PGP SIGNATURE-----
(none) /mnt/gentoo # _
```

可以看到完全一致。

3.解压Stage3

```
tar xpvf stage3-*.tar.xz --xattrs-include='*.*' --numeric-owner
```

`--numeric-owner` 被用于确保从tarball中提取的文件的用户和组ID与Gentoo发布工程团队预期的保持一致。

4.配置编译选项

编辑make.conf文件，优化编译参数：

```
nano -w /mnt/gentoo/etc/portage/make.conf
```

改成如下内容:

```
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /usr/share/portage/config/make.conf.example for a more
# detailed example.
COMMON_FLAGS="-march=native -O2 -pipe"
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
FCFLAGS="${COMMON_FLAGS}"
FFLAGS="${COMMON_FLAGS}"

MAKEOPTS="-j16"
# NOTE: This stage was built with the bindist Use flag enabled

# This sets the language of build output to English.
# Please keep this setting intact when reporting bugs.
LC_MESSAGES=C
```

其中 `MAKEOPTS` 表示编译时使用的线程数量，官方的建议是在 `CPU的线程数` 和 `整个系统的内存 / 2GiB` 中选择小的那个比较好，我这里设置为16。

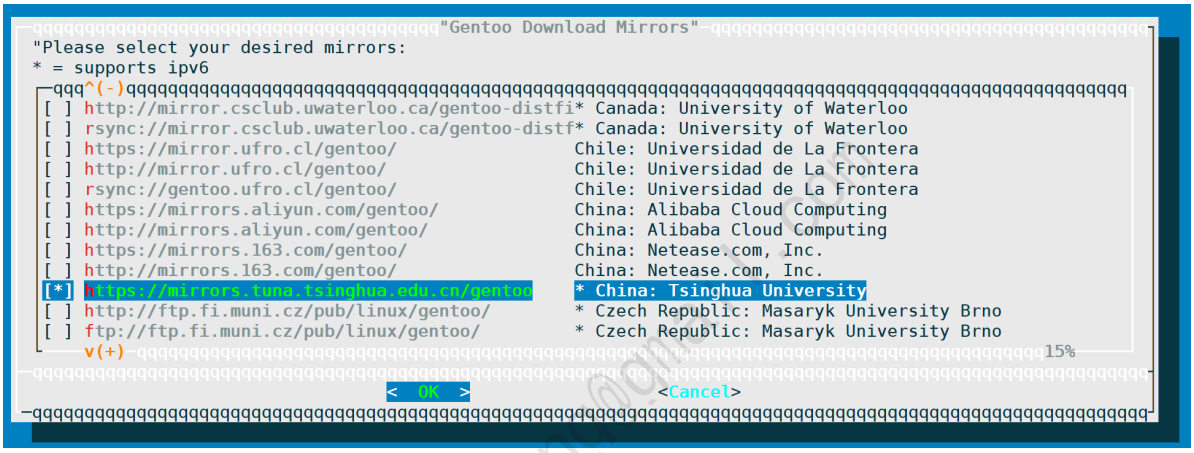
六、安装Gentoo基础系统

1.选择镜像站点

Gentoo默认配置的是官方源，跨境传输速度较慢，这里建议配置为国内源，比如选择清华源或阿里、网易等多个软件源。

```
mirrorselect -i -o >> /mnt/gentoo/etc/portage/make.conf
```

空格勾选即可：



2.设置Gentoo ebuild 软件仓库

Portage为Gentoo的包管理器，这里需要创建Portage配置目录，并将Gentoo镜像的配置文件复制到创建的 `repos.conf` 目录：

```
mkdir --parents /mnt/gentoo/etc/portage/repos.conf
cp /mnt/gentoo/usr/share/portage/config/repos.conf
/mnt/gentoo/etc/portage/repos.conf/gentoo.conf
```

验证是否成功，可以看下文件内容：

```
cat /mnt/gentoo/etc/portage/repos.conf/gentoo.conf
```

```
(none) /mnt/gentoo # mkdir --parents /mnt/gentoo/etc/portage/repos.conf
(none) /mnt/gentoo # cp /mnt/gentoo/usr/share/portage/config/repos.conf /mnt/gentoo/etc/portage/repos.conf/gentoo.conf
(none) /mnt/gentoo # cat /mnt/gentoo/etc/portage/repos.conf/gentoo.conf
[DEFAULT]
main-repo = gentoo

[gentoo]
location = /var/db/repos/gentoo
sync-type = rsync
sync-uri = rsync://rsync.gentoo.org/gentoo-portage
auto-sync = yes
sync-rsync-verify-jobs = 1
sync-rsync-verify-metamanifest = yes
sync-rsync-verify-max-age = 24
sync-openpgp-key-path = /usr/share/openpgp-keys/gentoo-release.asc
sync-openpgp-keyserver = hkps://keys.gentoo.org
sync-openpgp-key-refresh-retry-count = 40
sync-openpgp-key-refresh-retry-overall-timeout = 1200
sync-openpgp-key-refresh-retry-delay-exp-base = 2
sync-openpgp-key-refresh-retry-delay-max = 60
sync-openpgp-key-refresh-retry-delay-mult = 4
sync-webrsync-verify-signature = yes
(none) /mnt/gentoo #
```

3.复制DNS信息

```
cp --dereference /etc/resolv.conf /mnt/gentoo/etc/
```

`--dereference`，可以保障如果`/etc/resolv.conf`是一个符号链接的话，复制的是`resolv.conf`的目标文件而不是这个链接文件。

4.挂载文件系统

想要确保新环境(`/mnt/gentoo`)能正常工作，需要把Live系统的一些文件系统挂载到`/mnt/gentoo`，才能确保能正常使用。

```
mount --types proc /proc /mnt/gentoo/proc
mount --rbind /sys /mnt/gentoo/sys
mount --make-rslave /mnt/gentoo/sys
mount --rbind /dev /mnt/gentoo/dev
mount --make-rslave /mnt/gentoo/dev
mount --bind /run /mnt/gentoo/run
mount --make-slave /mnt/gentoo/run
```

5.chroot

使用`chroot`命令进入到Gentoo环境：

```
chroot /mnt/gentoo /bin/bash #进入到gentoo基础系统，并且使用bash解释器
source /etc/profile  #是配置文件重新生效
export PS1="(chroot) ${PS1}"  #修改PS1，帮助我们记忆这是在chroot环境
```

```
(none) /mnt/gentoo # chroot /mnt/gentoo /bin/bash
(none) / # source /etc/profile
(none) / # export PS1="(chroot) ${PS1}"
(chroot) (none) / #
```

6.挂载boot分区

安装内核及引导程序时需用到boot，因此需要提前挂载：

```
mount /dev/sda1 /boot
```

```
(chroot) (none) / # mount /dev/sda1 /boot/
(chroot) (none) / # df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/sda3      ext4       96G  1.5G   90G   2% /
cgroup_root    tmpfs      10M     0   10M   0% /sys/fs/cgroup
udev           devtmpfs   10M     0   10M   0% /dev
tmpfs          tmpfs      16G     0   16G   0% /dev/shm
none           tmpfs      16G  784K   16G   1% /run
/dev/sda1      vfat      253M   512  253M   1% /boot
(chroot) (none) / #
```

7.配置Portage

1）安装Gentoo ebuild数据库快照

```
emerge-webrsync
```


2）更新Portage ebuild数据库(可选)

上一步已经建立数据库快照了并且更新ebuild为了最新版本，如果距离上一步有停顿一段时间(几个小时或者一天)，可以使用下面命令保持为最新：

```
emerge --sync
```

3）eselect news

当更新Portage ebuild数据库后，Portage可能会输出类似下面的信息：

```
 * IMPORTANT: 9 news items need reading for repository 'gentoo'.
 * Use eselect news read to view new items.
```

一般都是推送的重要消息，可以使用eselect命令来阅读，不然每次emerge安装软件时都会在最后提示此信息。

```
eselect news list
eselect news read
```

4）选择正确的profile

使用如下命令查看目前选择的profile：

```
eselect profile list
```

```
(chroot) (none) / # eselect profile list
Available profile symlink targets:
  [1]   default/linux/amd64/17.1 (stable)
  [2]   default/linux/amd64/17.1/selinux (stable)
  [3]   default/linux/amd64/17.1/hardened (stable)
  [4]   default/linux/amd64/17.1/hardened/selinux (stable)
  [5]   default/linux/amd64/17.1/desktop (stable)
  [6]   default/linux/amd64/17.1/desktop/gnome (stable)
  [7]   default/linux/amd64/17.1/desktop/gnome/systemd (stable)
  [8]   default/linux/amd64/17.1/desktop/gnome/systemd/merged-usr (stable)
  [9]   default/linux/amd64/17.1/desktop/plasma (stable)
  [10]  default/linux/amd64/17.1/desktop/plasma/systemd (stable)
  [11]  default/linux/amd64/17.1/desktop/plasma/systemd/merged-usr (stable)
  [12]  default/linux/amd64/17.1/desktop/systemd (stable)
  [13]  default/linux/amd64/17.1/desktop/systemd/merged-usr (stable)
  [14]  default/linux/amd64/17.1/developer (exp)
  [15]  default/linux/amd64/17.1/no-multilib (stable)
  [16]  default/linux/amd64/17.1/no-multilib/hardened (stable)
  [17]  default/linux/amd64/17.1/no-multilib/hardened/selinux (stable)
  [18]  default/linux/amd64/17.1/no-multilib/systemd (dev)
  [19]  default/linux/amd64/17.1/no-multilib/systemd/merged-usr (dev)
  [20]  default/linux/amd64/17.1/no-multilib/systemd/selinux (exp)
  [21]  default/linux/amd64/17.1/no-multilib/systemd/selinux/merged-usr (exp)
  [22]  default/linux/amd64/17.1/systemd (stable) *
  [23]  default/linux/amd64/17.1/systemd/merged-usr (stable)
  [24]  default/linux/amd64/17.1/systemd/selinux (exp)
  [25]  default/linux/amd64/17.1/systemd/selinux/merged-usr (exp)
  [26]  default/linux/amd64/17.1/clang (exp)
  [27]  default/linux/amd64/17.1/systemd/clang (exp)
  [28]  default/linux/amd64/17.1/systemd/clang/merged-usr (exp)
  [29]  default/linux/amd64/17.0/x32 (dev)
  [30]  default/linux/amd64/17.0/x32/systemd (exp)
```

最后面 `*` 号表示当前选的配置文件。选的时候不能跨版本选择，比如17.1就选17.1的profile，不要17.0、17.1都包含。

因为我们使用最小化安装，这里默认就行，系统已经帮我们选好，如果是OpenRC Stage3包则默认会选OpenRC的profile(序列号1)。

比如选择OpenRC：

```
eselect profile set 1
```

5）更新@world集合

@world集合是Gentoo所有软件、配置的集合，进行构建系统之前，更新一下**@world集合**是必要的：

```
emerge --ask --verbose --update --deep --newuse @world
```

6）配置USE变量

USE变量可以在编译时选择编译或者不编译某些可选的支持、扩展包，简单理解为启用或禁用主程序某些其他功能，这样可以做到按需安装，只安装我们想要的包，更加精简化。

比如有以下需求：需要安装nmap，但是nmap的USE变量默认没有nping、ncat子命令，我们需要把它们一并安装上。

每个包的USE定义都可以在/etc/portage/package.use目录下设置，它可能也是一个文件，不管是哪一种，语法都是一样。

比如基于如上需求，需要安装nping、ncat子命令，那么USE变量可以是：

```
net-analyzer/nmap ipv6 nse ssl -libssh2 ncat nping -symlink (-system-lua) -verify-sig
```

```
 ⊙ 11:39:57  ~  cat /etc/portage/package.use/nmap
net-analyzer/nmap ipv6 nse ssl -libssh2 ncat nping -symlink (-system-lua) -verify-sig
 ⊙ 11:40:56  ~
```

要看nmap支持哪些USE标记，通过 `emerge -pv nmap` 即可查看：

```
emerge -pv nmap
```

因为配置了nmap的USE设置，所以默认会读取设置后的USE变量：

```
 ⊙ 11:42:10  ~  emerge -pv nmap                                                                    800  0.00  17.01.23

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R    ] net-analyzer/nmap-7.93-r2::gentoo  USE="ipv6 ncat nping nse ssl -libssh2 -symlink (-system-lua) -verify-sig" LUA_SINGLE_TARGET="lua5-3" 0 KiB

Total: 1 package (1 reinstall), Size of downloads: 0 KiB
 ⊙ 11:43:19  ~  _                                                                                 800  0.00  17.01.23
```

`-` 开头表示不安装，比如 `libssh2` 不安装 `libssh2` 库文件。

默认的USE设置在make.defaults文件中，可以使用如下命令查看当前USE标记：

```
emerge --info |grep ^USE
```

如果对USE不了解，这里建议保持默认。

同时可以在 `/var/db/repos/gentoo/profiles/use.desc` 中找到可用的USE标记的完整描述：

```
less /var/db/repos/gentoo/profiles/use.desc
```

以下8、9点根据当前init系统做对应操作，后续操作步骤会多次用到文本编辑器，如果不习惯内置的 `nano` 可以通过 `emerge --ask vim` 安装vim命令。

8.使用SystemD作为init系统

1）软链/etc/mtab

```
ln -sf /proc/self/mounts /etc/mtab
```

上游仅支持 `/etc/mtab`，如果不建立此软连接，则会导致 `mount` 、 `df` 命令出现bug(**#434090**和#**#477240**)。

2）生成Machine ID

创建一个随机机器ID：

```
systemd-machine-id-setup
```

3）设置主机名

```
hostnamectl set-hostname <HOSTNAME>
```

4）设置时区

时区列表在 `/usr/share/zoneinfo` 目录下，查到对应可用的时区后，软链到 `/etc/localtime` 。

比如我们要设置到上海时区：

```
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

```
(chroot) (none) / # ls /usr/share/zoneinfo
Africa      Atlantic   Canada  EST5EDT  Factory  GMT-0      Iceland  Japan      MST7MDT  PRC       ROC       US         Zulu            zone.tab
America     Australia  Chile   Egypt    GB       GMT0       Indian   Kwajalein  Mexico   PST8PDT   ROK       UTC        iso3166.tab     zone1970.tab
Antarctica  Brazil     Cuba    Eire     GB-Eire  Greenwich  Iran     Libya      NZ       Pacific   Singapore Universal  leap-seconds.list
Arctic      CET        EET     Etc      GMT      HST        Israel   MET        NZ-CHAT  Poland    Turkey    W-SU       leapseconds
Asia        CST6CDT    EST     Europe   GMT+0    Hongkong   Jamaica  MST        Navajo   Portugal  UCT       WET        tzdata.zi
(chroot) (none) / # ln -sf /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime
(chroot) (none) / #
```

5）设置语言环境

```
vim /etc/locale.conf
```

习惯使用英文环境，修改成 `LANG="en_US.utf8"` 即可。

9.使用OpenRC作为init系统

1）设置时区

依然以上海为例：

```
echo 'Asia/Shanghai' > /etc/timezone
```

重新配置timezone-data：

```
emerge --config sys-libs/timezone-data
```

将会基于 `/etc/timezone` 去更新 `/etc/localtime` 文件， `/etc/localtime` 会让系统的C类库知道系统在什么时区。

10.配置区域设置

1）生成区域设置

完整的locale支持列表可以在 `/usr/share/i18n/SUPPORTED` 文件找到。系统支持的locale必须在 `/etc/local.gen` 定义。

修改 `/etc/locale.gen` 文件，将前面的注释去掉，启用US字符格式：

```
en_US.UTF-8 UTF-8
```

运行 `locale-gen` ，将会生成 `/etc/locale.gen` 文件中指定的地区：

```
locale-gen
```

可以使用 `locale -a` 验证当前选择的 `locale` ：

```
(chroot) (none) / # locale-gen
 * Generating 2 locales (this might take a while) with 64 jobs
 *  (2/2) Generating C.UTF-8 ...
 *  (1/2) Generating en_US.UTF-8 ...
 * Generation complete
 * Adding locales to archive ...
(chroot) (none) / # locale -a
C
C.utf8
POSIX
en_US.utf8
(chroot) (none) / # _
```


2）选择区域设置

通过如下命令显示可用的配置：

```
eselect locale list
```

```
(chroot) (none) / # eselect locale list
Available targets for the LANG variable:
  [1]   C
  [2]   C.utf8
  [3]   POSIX
  [4]   en_US.utf8 *
  [ ]   (free form)
(chroot) (none) / # _
```

可以看到，系统默认已经帮我们选择了 `en_US.utf8` ，如果不符合预期，可通过 `eselect locale set <NUMBER>` 选择正确的区域设置，比如选择 `POSIX` ，可以是：

```
eselect locale set 3
```

完成上述选择后，重新加载环境：

```
env-update && source /etc/profile && export PS1="(chroot) ${PS1}"
```

```
(chroot) (none) / # env-update && source /etc/profile && export PS1="(chroot) ${PS1}"
>>> Regenerating /etc/ld.so.cache...
(chroot) (none) / # _
```

七、安装内核

以下内核安装方式二选一进行，ESXi上运行则更建议使用第一种。

1.安装二进制内核

ESXi安装方式最推荐使用二进制内核，如果你不厌其烦要从源码编译内核，那就需要把vmware虚拟化、相关驱动等诸多内核配置查找出来并勾选，费时费力，最后还不一定能完美兼容。

使用如下命令安装二进制内核：

```
emerge --ask sys-kernel/gentoo-kernel-bin
```

```
(chroot) (none) / # emerge --ask sys-kernel/gentoo-kernel-bin

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N     ] app-text/asciidoc-10.2.0  PYTHON_SINGLE_TARGET="python3_10 (-pypy3) -python3_9 -python3_11"
[ebuild  N     ] dev-libs/elfutils-0.188  USE="bzip2 nls utils -lzma -static-libs -test -valgrind -verify-sig -zstd" ABI_X86="(64) -32 (-x
32)"
[ebuild  N     ] virtual/libelf-3-r1  ABI_X86="(64) -32 (-x32)"
[ebuild  N     ] app-arch/cpio-2.13-r5  USE="nls"
[ebuild  N     ] app-alternatives/cpio-0  USE="gnu (split-usr) -libarchive"
[ebuild  N     ] sys-kernel/dracut-057-r3  USE="(-selinux) (-test)"
[ebuild  N     ] sys-kernel/gentoo-kernel-bin-5.15.85-r1  USE="initramfs -test"
[ebuild  N     ] virtual/dist-kernel-5.15.85

Would you like to merge these packages? [Yes/No]
```

可以看到当前源里面最新stable版本为5.15.85-r1，且默认USE了 `initramfs` 。

安装成功后，可以通过 `eselect` 命令看到当前版本：

```
eselect kernel list
```

```
(chroot) (none) / # eselect kernel list
Available kernel symlink targets:
  [1]   linux-5.15.85-gentoo-dist *
(chroot) (none) / # _
```

2.源码编译内核

1）安装内核源码

```
emerge --ask sys-kernel/gentoo-sources
```

列出已安装的内核：

```
eselect kernel list
```

```
(chroot) (none) / # eselect kernel list
Available kernel symlink targets:
  [1]   linux-5.15.80-gentoo
  [2]   linux-5.15.85-gentoo-dist *
(chroot) (none) / #
```

可以看到一共有两个内核，第一个是从源码安装的版本，第二个是刚安装的二进制版本。

如果没有进行二进制内核安装，这里只会看到一个内核，不需要做额外操作。因为这里有两个内核，且第一是源码安装的内核版本，我们选择第一个：

```
eselect kernel set 1
```

```
(chroot) (none) / # eselect kernel set 1
(chroot) (none) / # ls -l /usr/src/linux
lrwxrwxrwx 1 root root 20 Jan 18 00:25 /usr/src/linux -> linux-5.15.80-gentoo
(chroot) (none) / #
```

可以看到，设置为第一个时，其实就是创建一个 `/usr/src/linux` 的链接指向内核目录。

2）手动配置内核

首先可以安装 `sys-apps/pciutils` 包，通过lspci命令来收集硬件信息：

```
emerge --ask sys-apps/pciutils
lspci
```

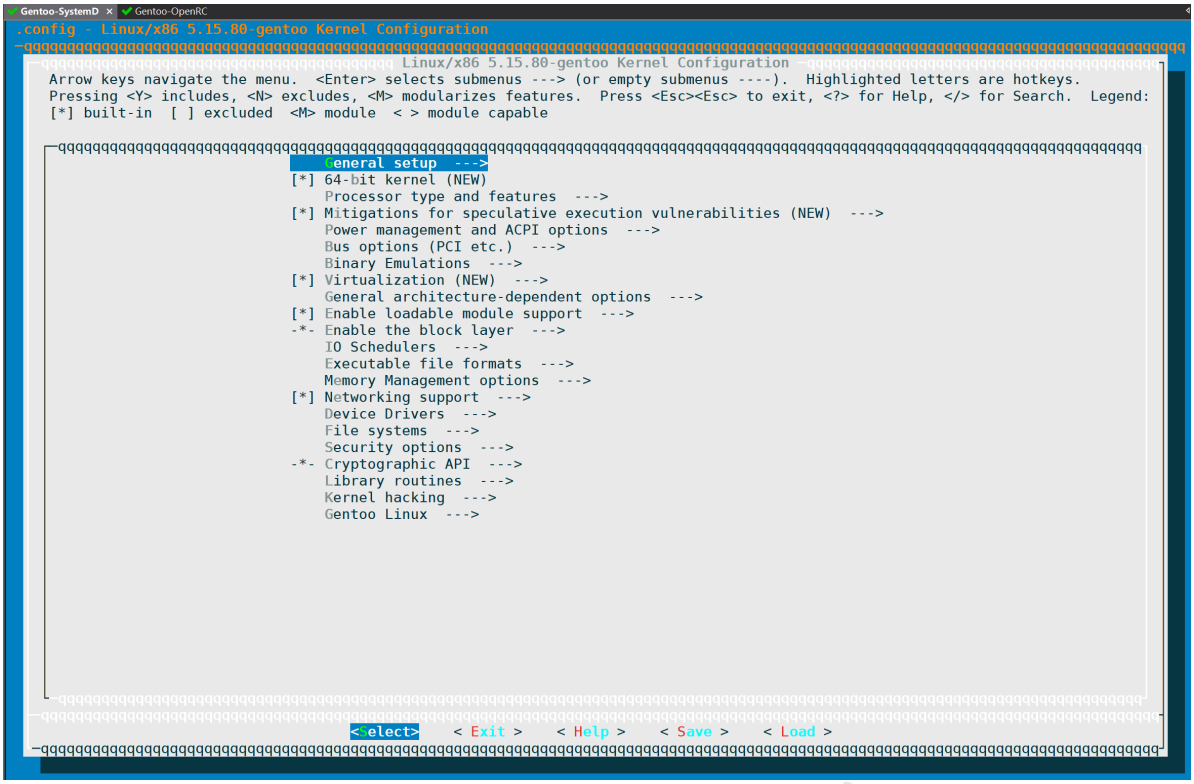
```
(chroot) (none) / # lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:07.7 System peripheral: VMware Virtual Machine Communication Interface (rev 10)
00:0f.0 VGA compatible controller: VMware SVGA II Adapter
00:11.0 PCI bridge: VMware PCI bridge (rev 02)
00:15.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.7 PCI bridge: VMware PCI Express Root Port (rev 01)
03:00.0 Serial Attached SCSI controller: VMware PVSCSI SCSI Controller (rev 02)
0b:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)
(chroot) (none) / #
```

因为下一步配置需要把每个必要的内核模块勾选。

接下来进入内核源码，开始配置内核：

```
cd /usr/src/linux
make menuconfig
```

一如既往的dialog视图：



接下来就是最繁琐的步骤，勾选必要的内核模块(空格勾选，*代表勾选上):

① 启用Gentoo特有选项

```
Gentoo Linux --->
  Generic Driver Options --->
    [*] Gentoo Linux support
    [*]   Linux dynamic and persistent device naming (userspace devfs) support
    [*]   Select options required by Portage features
        Support for init systems, system and service managers  --->
          [*] OpenRC, runit and other script based systems and managers
          [*] systemd
```

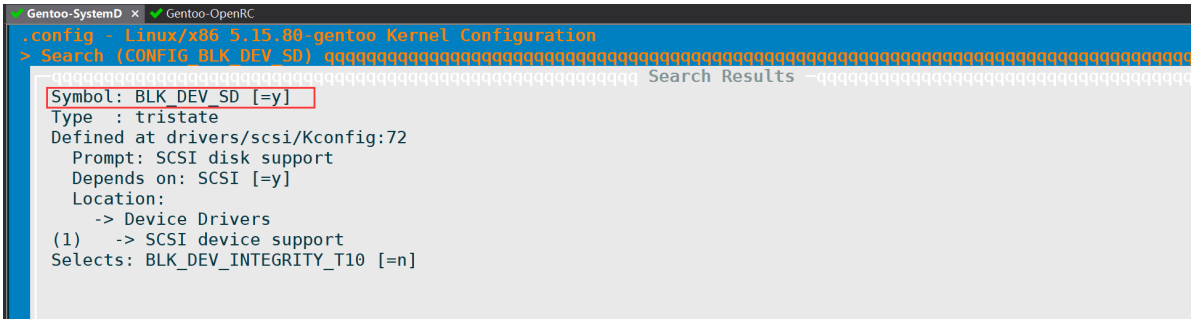
② 启用devtmpfs支持

```
Device Drivers --->
  Generic Driver Options --->
    [*] Maintain a devtmpfs filesystem to mount at /dev
    [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
```

③ 启用SCSI硬盘支持

```
Device Drivers --->
   SCSI device support  --->
      <*> SCSI disk support
```

验证SCSI磁盘支持是否已激活(`CONFIG_BLK_DEV_SD`)，可以通过 `/` ，来搜索 `CONFIG_BLK_DEV_SD` ，这样不用一个个子选项去找，可以看到默认已经勾选:



④ 选择所需要的文件系统

```
File systems --->
  <*> Second extended fs support
  <*> The Extended 3 (ext3) filesystem
  <*> The Extended 4 (ext4) filesystem
  <*> Reiserfs support
  <*> JFS filesystem support
  <*> XFS filesystem support
  <*> Btrfs filesystem support
  DOS/FAT/NT Filesystems  --->
    <*> MSDOS fs support
    <*> VFAT (Windows-95) fs support

  Pseudo Filesystems --->
    [*] /proc file system support
    [*] Tmpfs virtual memory file system support (former shm fs)
```


⑤ 激活SMP支持

CPU多核需要用到，配置名为：`CONFIG_SMP`，这里默认也是勾选的，可以检查一遍：

```
Processor type and features  --->
  [*] Symmetric multi-processing support
```

⑥ 启用对GPT的支持

如果在分区时使用GPT分区，则需要启用它（`CONFIG_PARTITION_ADVANCED`，`CONFIG_EFI_PARTITION`）：

```
-*- Enable the block layer --->
   Partition Types --->
      [*] Advanced partition selection
      [*] EFI GUID Partition support
```

⑦ 启用对UEFI的支持

如果你用的是UEFI引导，则需要启用EFI stub支持（`CONFIG_EFI`，`CONFIG_EFI_STUB`，`CONFIG_EFI_MIXED`，`CONFIG_EFI_VARS`，`CONFIG_EFI_FB`）

```
Processor type and features  --->
    [*] EFI runtime service support
    [*]   EFI stub support
    [*]     EFI mixed-mode support

Device Drivers
    Firmware Drivers  --->
        EFI (Extensible Firmware Interface) Support  --->
            <*> EFI Variable Support via sysfs
    Graphics support  --->
        Frame buffer Devices  --->
            <*> Support for frame buffer devices  --->
                [*]   EFI-based Framebuffer Support
```

⑧ 启用对ESXi的支持

针对ESXi，**官方**给出了需要勾选的内核模块列表：

```
[*] Networking support  --->
        Networking options  --->
            <*> Virtual Socket protocol
            <*>   VMware VMCI transport for Virtual Sockets
    Device Drivers  --->
        [*] Fusion MPT device support  --->
                <*>   Fusion MPT ScsiHost drivers for SPI
            Misc devices  --->
                <*> VMware Balloon Driver
                <*> VMware VMCI Driver
            SCSI device support  --->
                [*] SCSI low-level drivers  --->
                   <*>   VMware PVSCSI driver support
        [*] Network device support  --->
                [*]   Ethernet driver support  --->
                    [*]   AMD devices
                    <*>     AMD 8111 (new PCI LANCE) support
                    <*>     AMD PCnet32 PCI support
                    [*]   Intel devices
                    <*>     Intel(R) PRO/1000 Gigabit Ethernet support
                    <*>     Intel(R) PRO/1000 PCI-Express Gigabit Ethernet support
                <*>   VMware VMXNET3 ethernet driver
            Input device support  --->
                [*]   Keyboards  --->
                    <*>   AT keyboard
    File systems  --->
        <*> FUSE (Filesystem in Userspace) support
```

配置文件名：

- CONFIG_NET_VENDOR_AMD
- CONFIG_AMD8111_ETH

- CONFIG_PCNET32
- CONFIG_NET_VENDOR_INTEL
- CONFIG_E1000
- CONFIG_E1000E
- CONFIG_KEYBOARD_ATKBD
- CONFIG_VMWARE_BALLOON
- CONFIG_VMWARE_PVSCSI
- CONFIG_VMXNET3
- CONFIG_VMWARE_VMCI
- CONFIG_VMWARE_VMCI_VSOCKETS
- CONFIG_FUSE_FS
- CONFIG_FUSION

同时可以考虑安装vm-tools工具：

```
emerge --ask app-emulation/open-vm-tools
```

3）编译和安装内核

```
make && make modules_install
```

无需用 `-j` 指定线程数，默认从我们前面的 `/etc/portage/make.conf` 读取 `MAKEOPTS` 指定的线程数。

编译完成后，使用如下命令完成安装，将会复制内核镜像到/boot目录：

```
make install
```

4）安装initramfs

安装前线安装 `dracut` ，用它来生成 `initramfs` ：

```
emerge --ask sys-kernel/dracut
dracut --kver=5.15.80-gentoo
```

成功后，可以在/boot目录下看到initramfs开头的文件：

```
(chroot) (none) /boot # ls initramfs*
initramfs-5.15.80-gentoo.img
(chroot) (none) /boot # _
```

八、配置系统

1.创建/etc/fstab

可以通过 `blkid` 命令查看磁盘UUID：

```
blkid
```

```
(chroot) (none) / # blkid
/dev/sr0: BLOCK_SIZE="2048" UUID="2023-01-01-18-30-54-00" LABEL="ISOIMAGE" TYPE="iso9660" PTTYPE="PMBR"
/dev/loop0: TYPE="squashfs"
/dev/sda2: UUID="e483307b-a5ce-44d2-94d6-d88b0584b5e0" TYPE="swap" PARTUUID="af2138c5-02"
/dev/sda3: UUID="7a05ddf2-b16c-4b4f-a70b-50288aededa8" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="af2138c5-03"
/dev/sda1: UUID="C8D9-EEED" BLOCK_SIZE="512" TYPE="vfat" PARTUUID="af2138c5-01"
(chroot) (none) / # _
```

拿到UUID后，写入到fstab里面：

```
vim /etc/fstab
```

> vim默认没装，可以使用 `emerge --ask app-editors/vim` 安装，要么就用 `nano` 命令。

比如如上UUID，写入为：

```
UUID=C8D9-EEED                              /boot           vfat
defaults,noatime 0 2
UUID=e483307b-a5ce-44d2-94d6-d88b0584b5e0   none            swap            sw
      0 0
UUID=7a05ddf2-b16c-4b4f-a70b-50288aededa8   /               ext4            noatime
     0 1
```

```
(chroot) (none) / # tail -n 3 /etc/fstab
UUID=C8D9-EEED                              /boot           vfat            defaults,noatime 0 2
UUID=e483307b-a5ce-44d2-94d6-d88b0584b5e0   none            swap            sw               0 0
UUID=7a05ddf2-b16c-4b4f-a70b-50288aededa8   /               ext4            noatime          0 1
(chroot) (none) / # _
```


2.配置主机名、域名

1）OpenRC

```
nano -w /etc/conf.d/hostname
```

2）SystemD

比如将主机名设置为gentoo：

```
hostnamectl hostname gentoo
```

3.配置网络

安装dhcpcd：

```
emerge --ask net-misc/dhcpcd
```

如果是OpenRC，默认启用此服务，让它开机自启动：

```
rc-update add dhcpcd default
rc-service dhcpcd start
```

SystemD则为：

```
systemctl enable dhcpcd
```

```
(chroot) (none) / # systemctl enable dhcpcd
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpcd.service → /lib/systemd/system/dhcpcd.service.
(chroot) (none) / # _
```

4.设置密码

不要忘了给系统设置一个密码，不然无法登录进去：

```
passwd
```

5.配置引导和启动

OpenRC无需执行此步骤，保持默认即可。

SystemD则需要通过如下两条命令，从Live环境平滑过渡到安装后的首次启动：

```
systemd-firstboot --prompt --setup-machine-id
systemctl preset-all
```

九、安装系统工具

1.安装日志工具

OpenRC可以安装它，如果是SystemD则不需要安装：

```
emerge --ask app-admin/sysklogd
```

在OpenRC上添加自启动：

```
rc-update add sysklogd default
```

2.安装Cron守护进程(可选)

```
emerge --ask sys-process/dcron
```

OpenRC上添加自启动

```
rc-update add dcron default
```

SystemD上添加自启动

```
systemctl enable dcron
```

初始化

```
crontab /etc/crontab
```

```
(chroot) (none) / # systemctl enable dcron
Created symlink /etc/systemd/system/multi-user.target.wants/dcron.service → /lib/systemd/system/dcron.service.
(chroot) (none) / # crontab /etc/crontab
(chroot) (none) / # _
```


3.文件索引(可选)

即locate工具，安装命令如下：

```
emerge --ask sys-apps/mlocate
```

4.远程访问(推荐)

sshd服务默认已经内置，我们只需要把它设置为开机自启动服务即可。

OpenRC

```
rc-update add sshd default
```

SystemD

```
systemctl enable sshd
```

需要注意，默认 `sshd_config` 禁用了root登录，可以考虑修改成允许：

```
sed -ir 's|^#(Permit.*) (pro.*)|\1 yes|g' /etc/ssh/sshd_config
```

之后重启sshd服务即可。

5.时间同步

使用NTP协议进行时钟同步的软件很多，比如ntpd或者chrony，这里建议使用后者：

```
emerge --ask net-misc/chrony
```

OpenRC

```
rc-update add chronyd default
```

SystemD

```
systemctl enable chronyd
```


6.安装文件系统工具

`/dev/sda1` 和 `/dev/sda3` 我们分别使用了FAT32和EXT4两个文件系统，所以至少需要安装这两个工具：

```
emerge --ask sys-fs/dosfstools sys-fs/e2fsprogs
```

十、安装及配置引导

以下选项根据实际引导对应选择

1.BIOS引导

当使用BIOS引导时，无需其他配置直接安装GRUB：

```
emerge --ask --verbose sys-boot/grub
```

在 `/etc/default/grub` 添加两行：

```
GRUB_CMDLINE_LINUX="init=/lib/systemd/systemd"
GRUB_DISABLE_OS_PROBER=false
```

安装grub到第一块盘：

```
grub-install /dev/sda
```

```
(chroot) (none) / # vim /etc/default/grub
(chroot) (none) / # grub-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
(chroot) (none) / #
```

生成 `grub.cfg` 配置文件：

```
grub-mkconfig -o /boot/grub/grub.cfg
```

```
(chroot) (none) / # grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.85-gentoo-dist
Found initrd image: /boot/initramfs-5.15.85-gentoo-dist.img
done
(chroot) (none) / # _
```


2.UEFI引导

同理，先安装grub：

```
emerge --ask --verbose sys-boot/grub
```

在将grub安装到引导分区：

```
grub-install --target=x86_64-efi --efi-directory=/boot
```

生成 `grub.cfg` 配置文件：

```
grub-mkconfig -o /boot/grub/grub.cfg
```

3.重启系统

以上步骤安装完成后，可以退出chroot环境，并使用umount取消挂载，最后重启系统：

```
(chroot) (none) / # exit
exit
(none) ~ # umount -l /mnt/gentoo/dev{/shm,/pts,}
(none) ~ # umount -R /mnt/gentoo
(none) ~ # reboot
```

```
(chroot) (none) / # exit
exit
(none) ~ # umount -l /mnt/gentoo/dev{/shm,/pts,}
(none) ~ # umount -R /mnt/gentoo
(none) ~ # reboot

Broadcast message from root@(none) (pts/0) (Tue Jan 17 18:02:18 2023):

The system is going down for reboot NOW!
```

重启后，可以正常进入系统，说明已经成功：

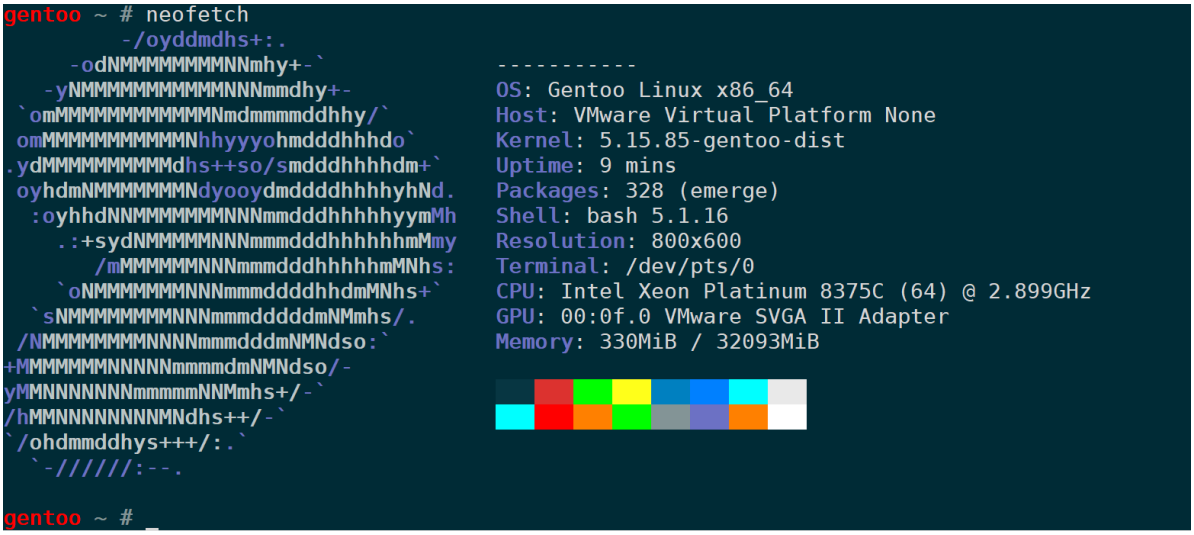
```
gentoo ~ # ifconfig
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.221  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::5b0:df6c:9f6:ef0f  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:81:41:27  txqueuelen 1000  (Ethernet)
        RX packets 642  bytes 56247 (54.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 112  bytes 18711 (18.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

gentoo ~ # _
```

进入系统后，stage3包我们已经不需要了，可以把它们删除掉：

```
rm /stage3-*.tar.*
```

同时可以安装 `neofetch` 工具看下系统概览：

```
emerge --ask app-misc/neofetch
```



到此，整个系统已经全部安装结束。