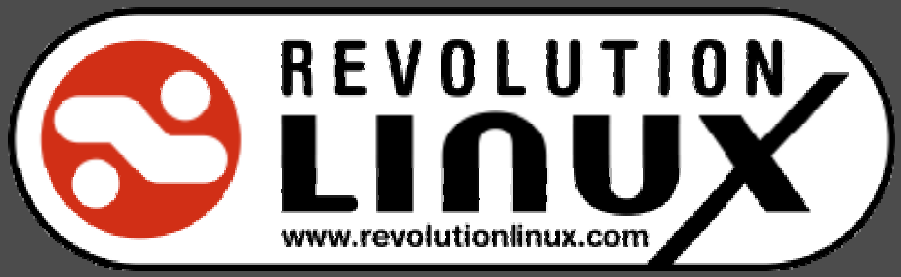
Automated OSCAR testing with Linux-VServers



Fernando Laudares Camargos
&
Benoît des Ligneris, Ph. D

fernando@revolutionlinux.com

Why ?

Why ?

- The testing process of a complex program managed with a version system and develop by a large team of international developers is a very lengthy and demanding task.

Why ?

- The testing process of a complex program managed with a version system and develop by a large team of international developers is a very lengthy and demanding task.

- An automated testing infrastructure for OSCAR that rely on the virtualization of Linux computers.

Why ?

- The testing process of a complex program managed with a version system and develop by a large team of international developers is a very lengthy and demanding task.

- An automated testing infrastructure for OSCAR that rely on the virtualization of Linux computers.

- The virtualization layer is provided by the Linux-VServer project

Why ?

- The testing process of a complex program managed with a version system and develop by a large team of international developers is a very lengthy and demanding task.

- An automated testing infrastructure for OSCAR that rely on the virtualization of Linux computers.

- The virtualization layer is provided by the Linux-VServer project

- Present the current design and implementation of the testing program from a developer's point of view.

Plan

Plan

- Introduction

Plan

- Introduction
- System overview

Plan

- Introduction
- System overview
- Actual implementation

Plan

- Introduction
- System overview
- Actual implementation
- Future contributions

Plan

- Introduction
- System overview
- Actual implementation
- Future contributions
- Conclusion

Introduction

Introduction

- The OSCAR project is maintained by a large group of developers that work in parallel while being geographically separated

Introduction

- The OSCAR project is maintained by a large group of developers that work in parallel while being geographically separated
- All the development efforts are centralized with the Subversion version system

Introduction

- While OSCAR support multiple distributions and multiple architectures, most of the OSCAR developers develop only on one one architecture and distribution, sometimes two.

Introduction

- ***While OSCAR support multiple distributions and multiple architectures, most of the OSCAR developers develop only on one one architecture and distribution, sometimes two.***

- ***This mean that a developer can not easily test the impact of his changes on another Linux distribution (architecture, version, variant) directly.***

Introduction

- As a consequence, tests are delayed until the last stages of publication of a new OSCAR version, when the subversion tree is frozen and OSCAR reaches the beta quality stage.

Introduction

- ***As a consequence, tests are delayed until the last stages of publication of a new OSCAR version, when the subversion tree is frozen and OSCAR reaches the beta quality stage.***

- Because of the relative inefficiency of this process, OSCAR releases has been less and less frequent: the burden of physically testing OSCAR against many architecture/distribution/version took a lot of developer's time that could be better used to develop new features and functionalities.

Introduction

- For those reasons is extremely important, for the evolution and stability of the OSCAR project, that a complete set of functional tests covering all the Linux distributions (supported) can be done periodically in a fully automated way.

Introduction

- For those reasons is extremely important, for the evolution and stability of the OSCAR project, that a complete set of functional tests covering all the Linux distributions (supported) can be done periodically in a fully automated way.

- However, such a procedure demands a complete assembling of an OSCAR cluster for each supported distribution, which consumes a huge amount of resources and time, making it a complex, and sometimes complicated, task.

Introduction (finally)

- *The automated testing should not replace completely the quality insurance process of the OSCAR project.*

Introduction (finally)

- The automated testing should not replace completely the quality insurance process of the OSCAR project.
- It has to be considered like a «simulation» of a cluster installation with OSCAR using only one physical computer.

Introduction (finally)

- *The automated testing should not replace completely the quality insurance process of the OSCAR project.*
- It has to be considered like a «simulation» of a cluster installation with OSCAR using only one physical computer.
- This being said, we believe that this testing infrastructure will allow the OSCAR project to «release early, release often» and, as a conseguence, to aument the functionality and general quality of the project.

Introduction (finally)

- The objective of this article is to present an automated testing infrastructure that will (**can**) be used for OSCAR development.

Introduction (finally)

- The objective of this article is to present an automated testing infrastructure that will (**can**) be used for OSCAR development.
- The main goal of this testing infrastructure is to reduce the development time used for testing the software.

Introduction (finally)

- ***The objective of this article is to present an automated testing infrastructure that will (can) be used for OSCAR development.***
- The main goal of this testing infrastructure is to reduce the development time used for testing the software.
- We took advantage of a stable and mature virtualization technology that is distribution independent and support several architectures.

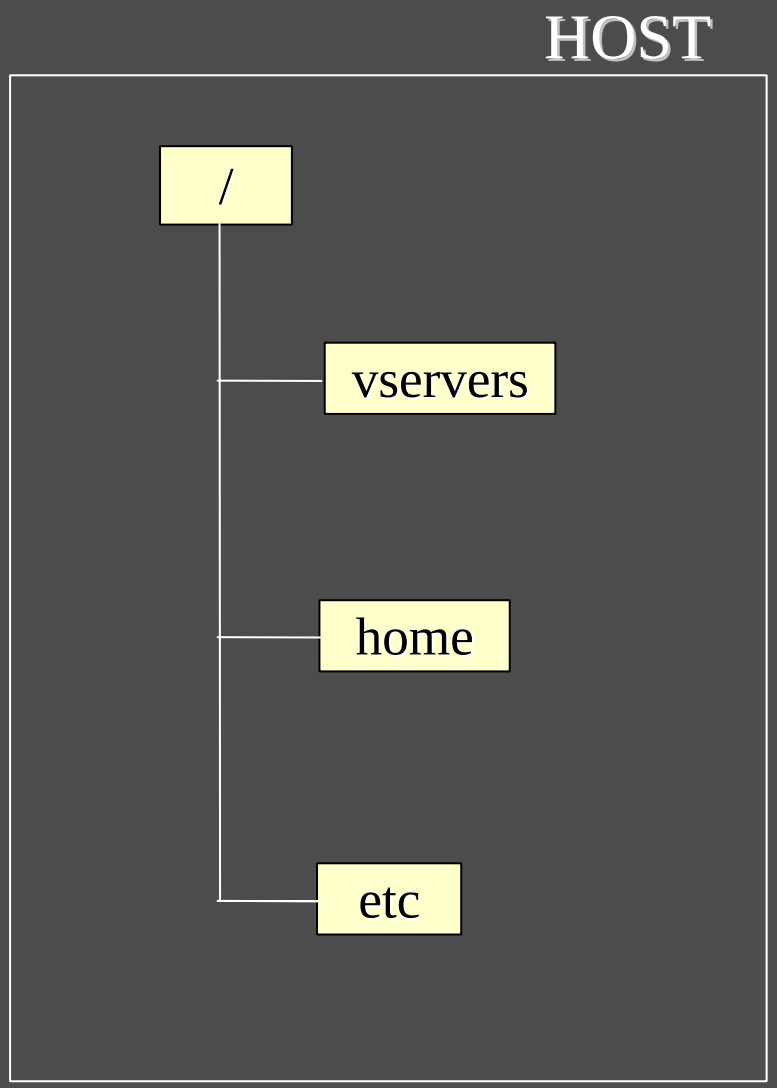
System overview

System overview

Before exposing the virtual testing infrastructure, we'll briefly present the real structure of the files and directories of the host system (Files organization), explain how the sources of OSCAR are «manipulated» (From suversion to tarball) and, finally, present the design utilized (Design overview)

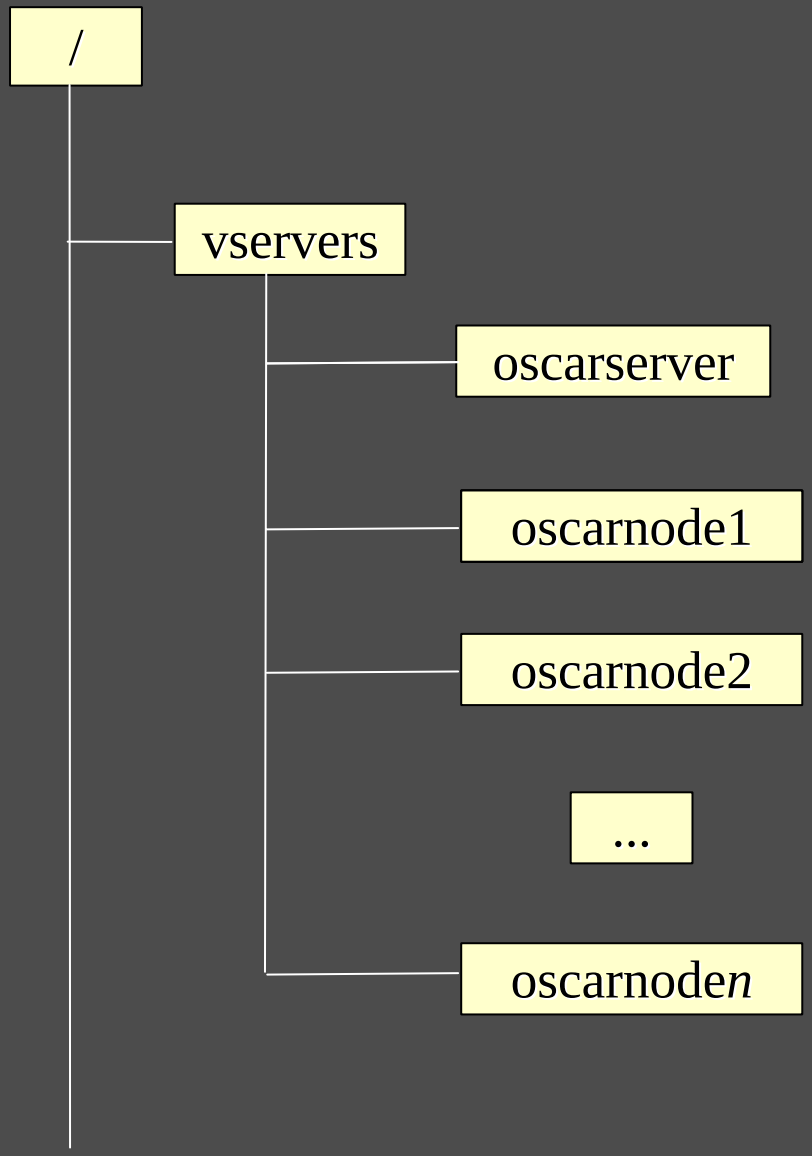
System overview

Files organization



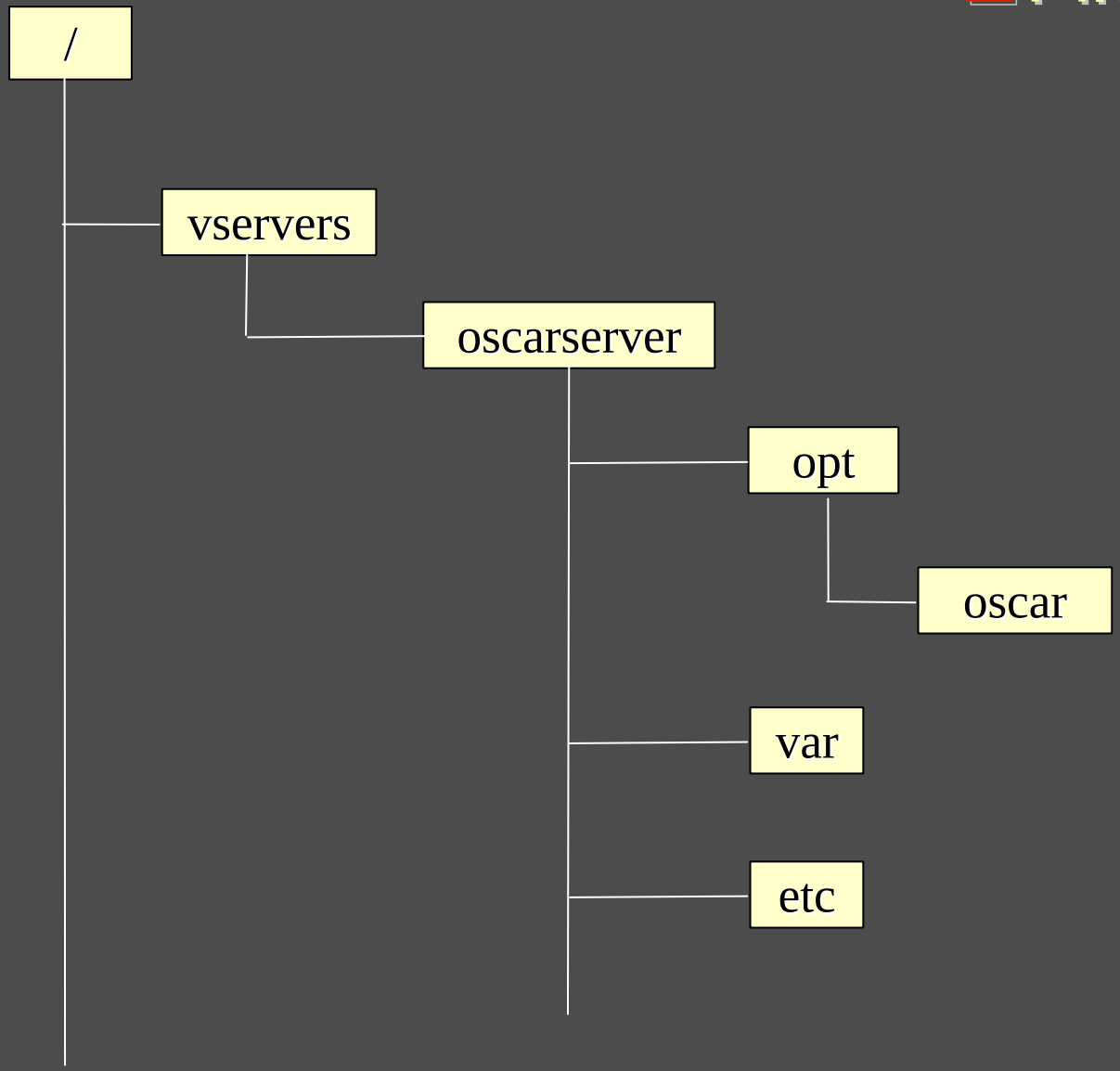
System overview

Files organization



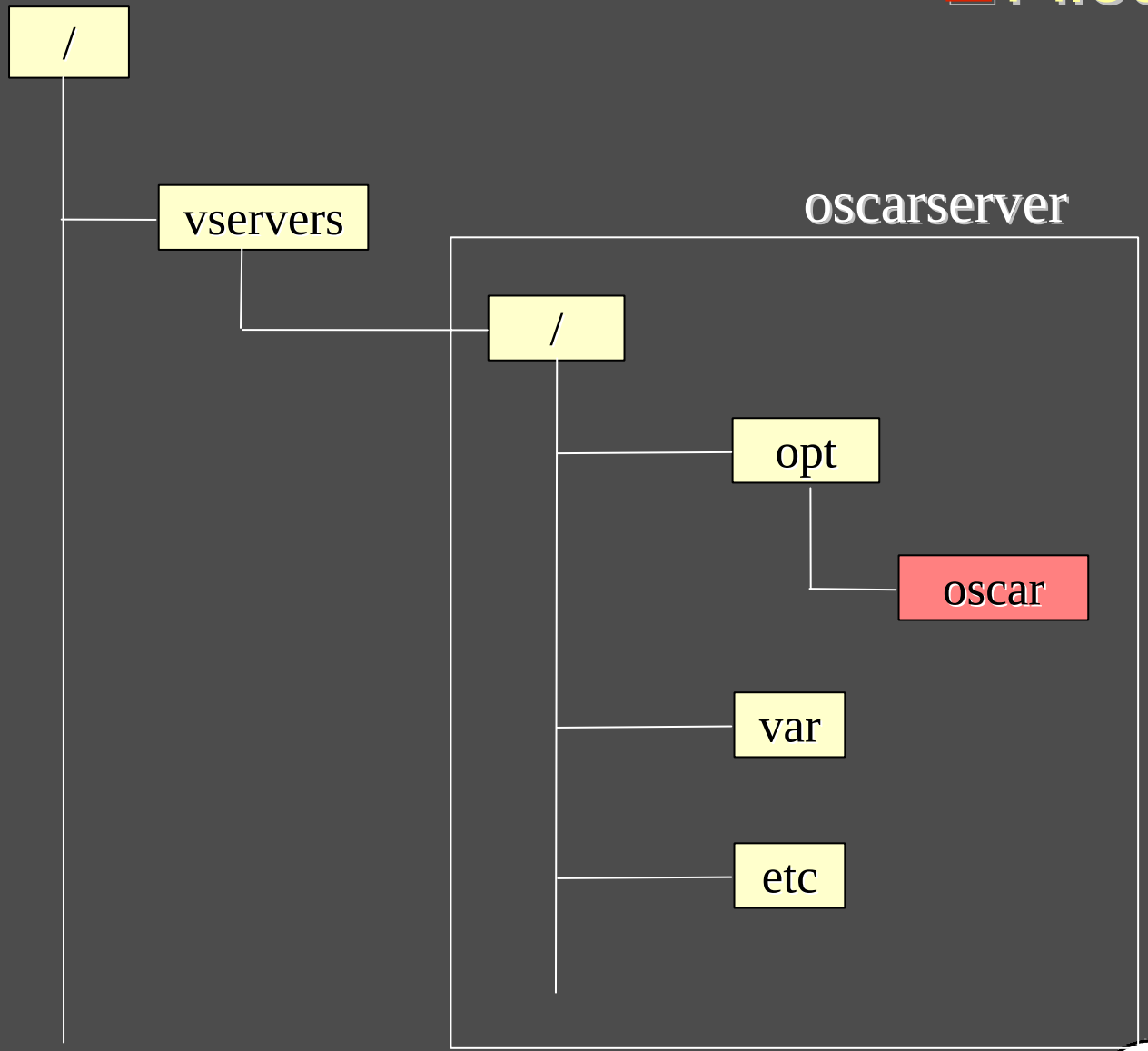
System overview

Files organization



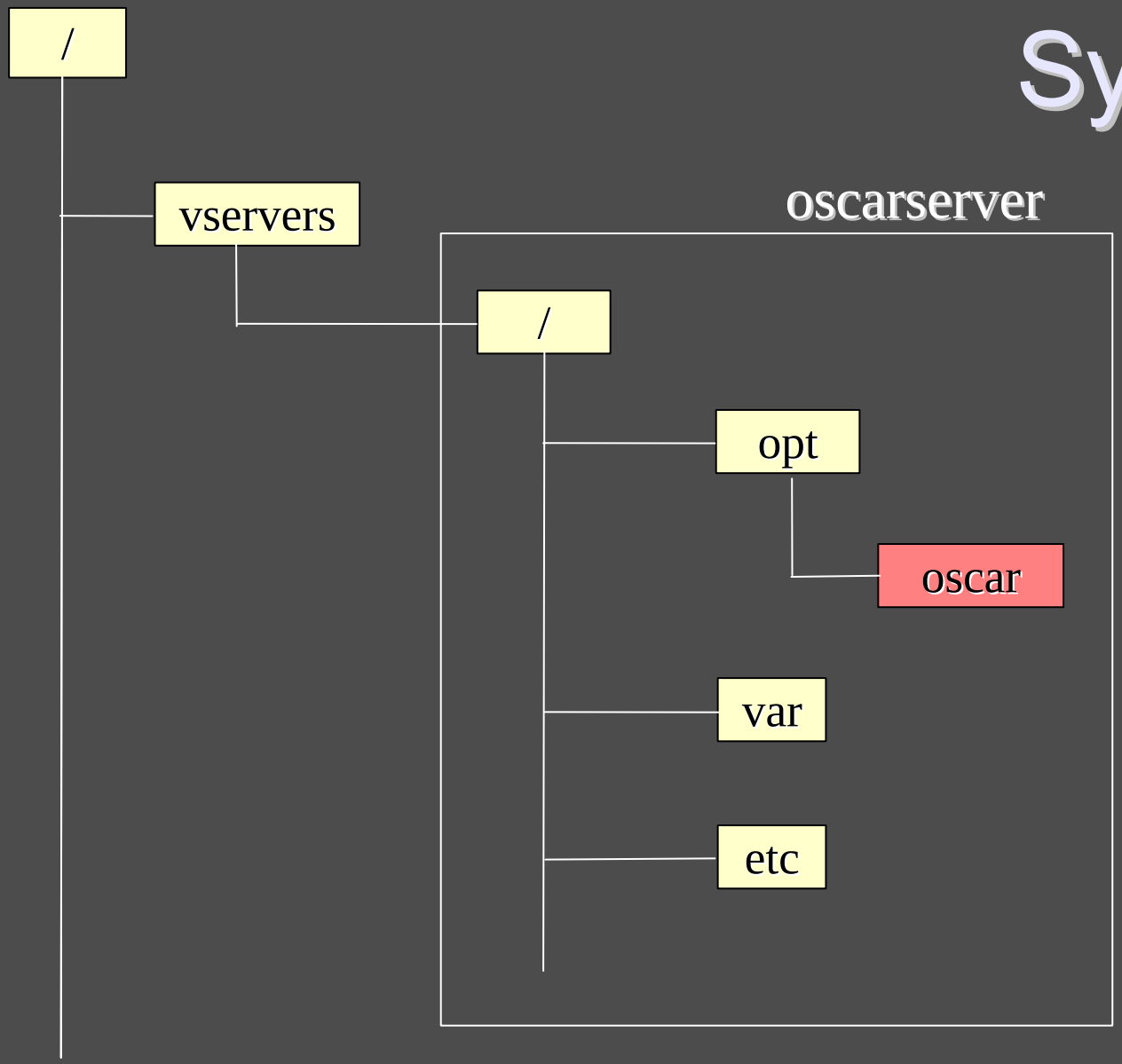
System overview

Files organization



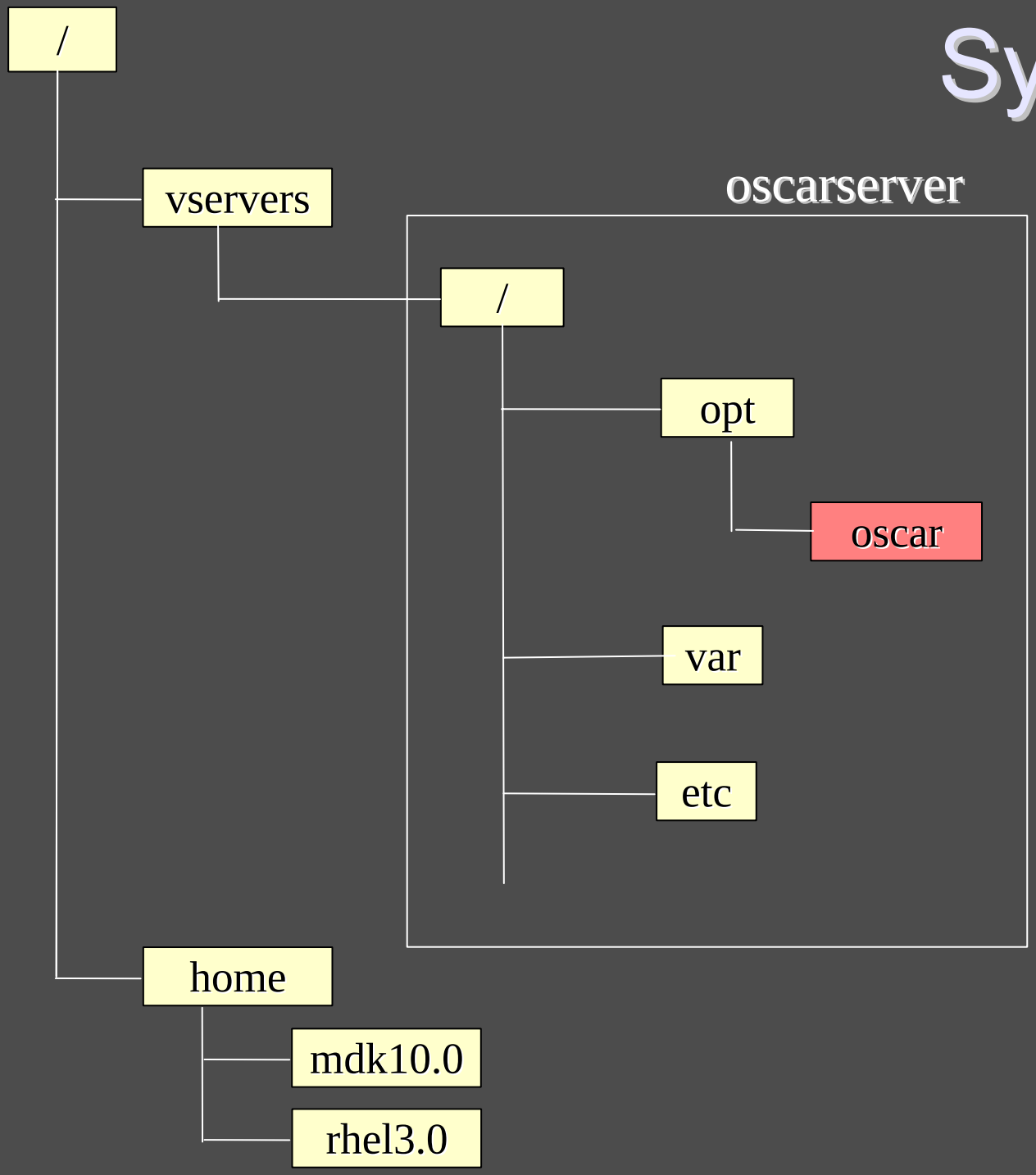
System overview

Files organization



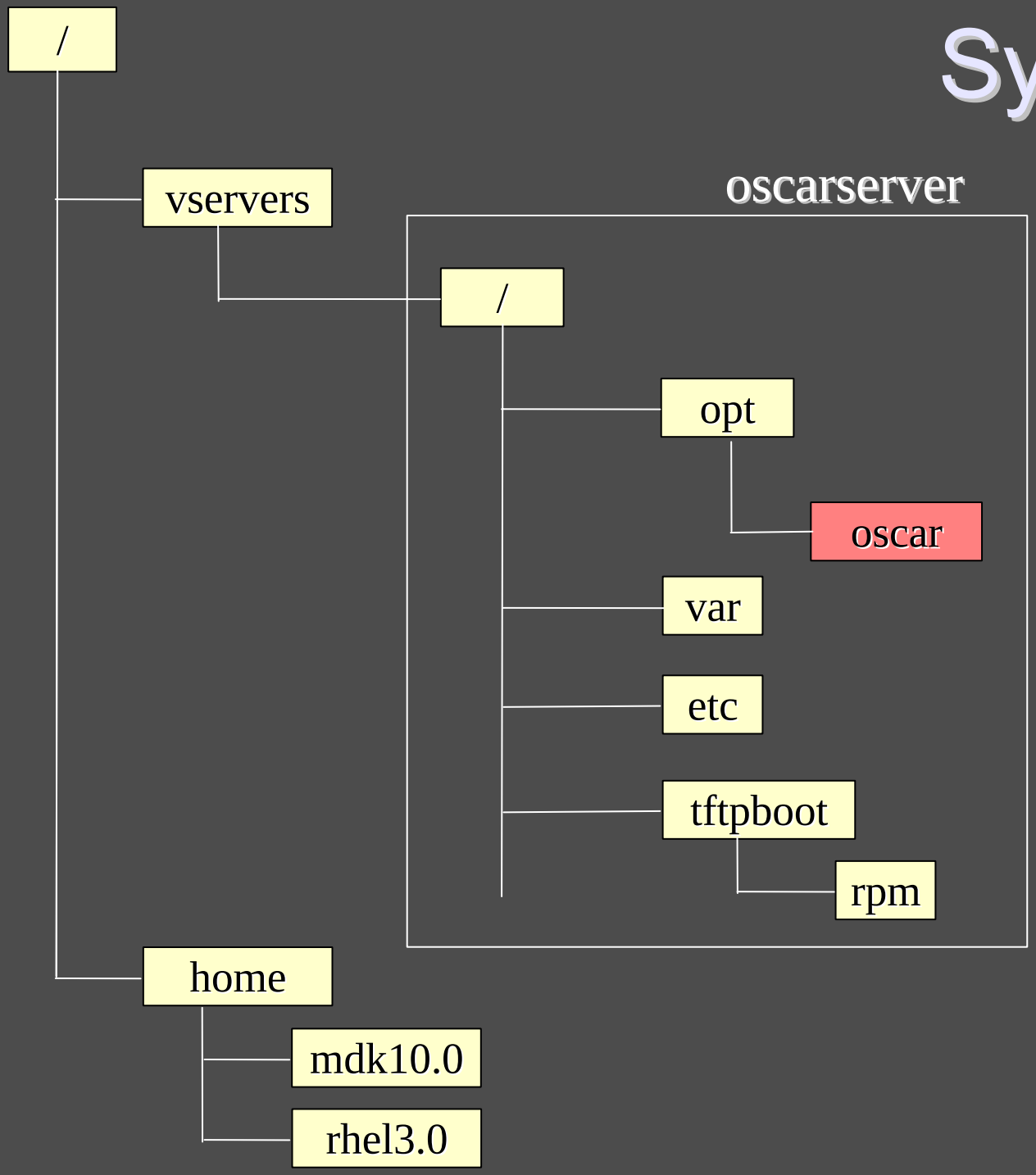
System overview

Files organization



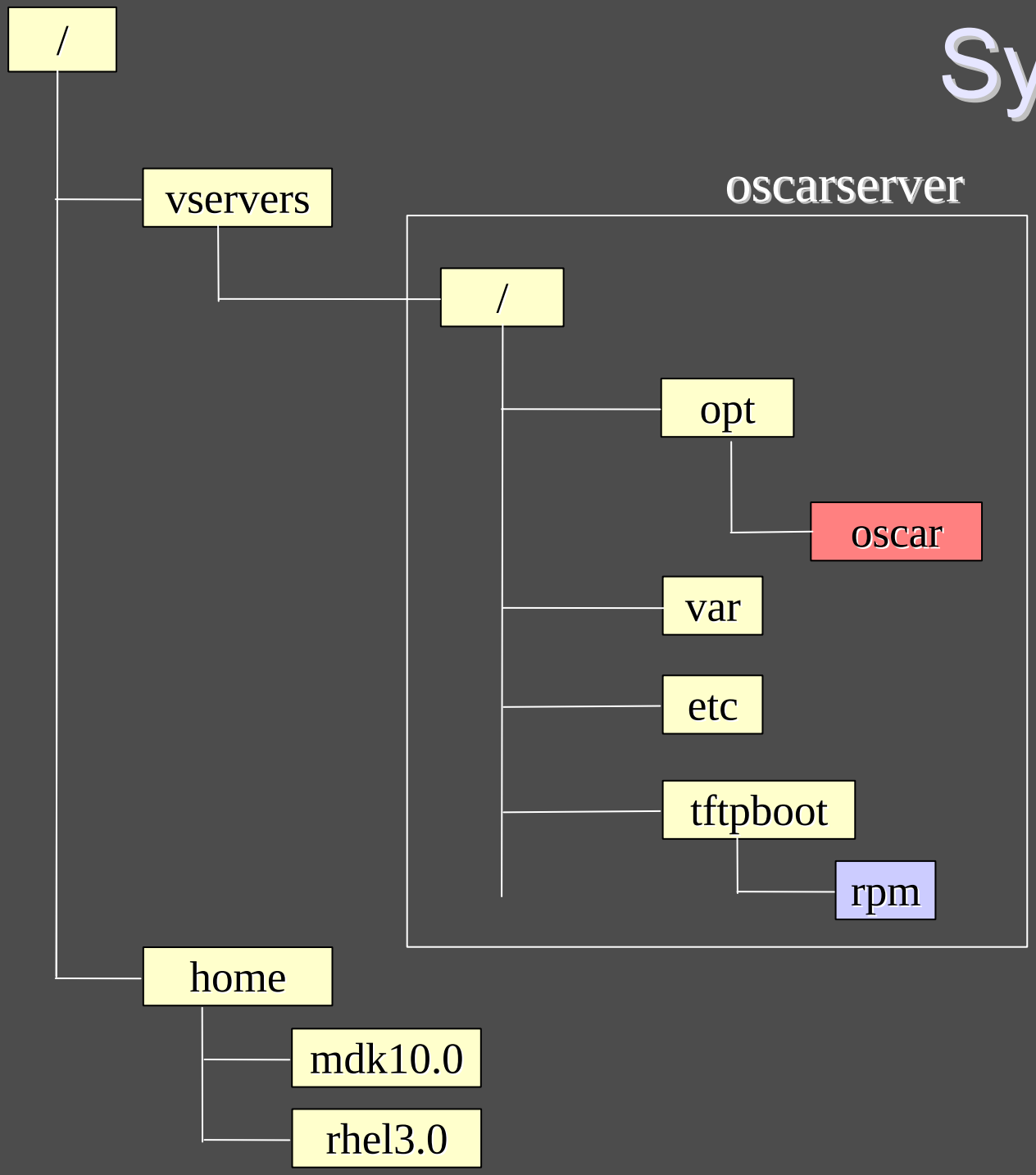
System overview

Files organization



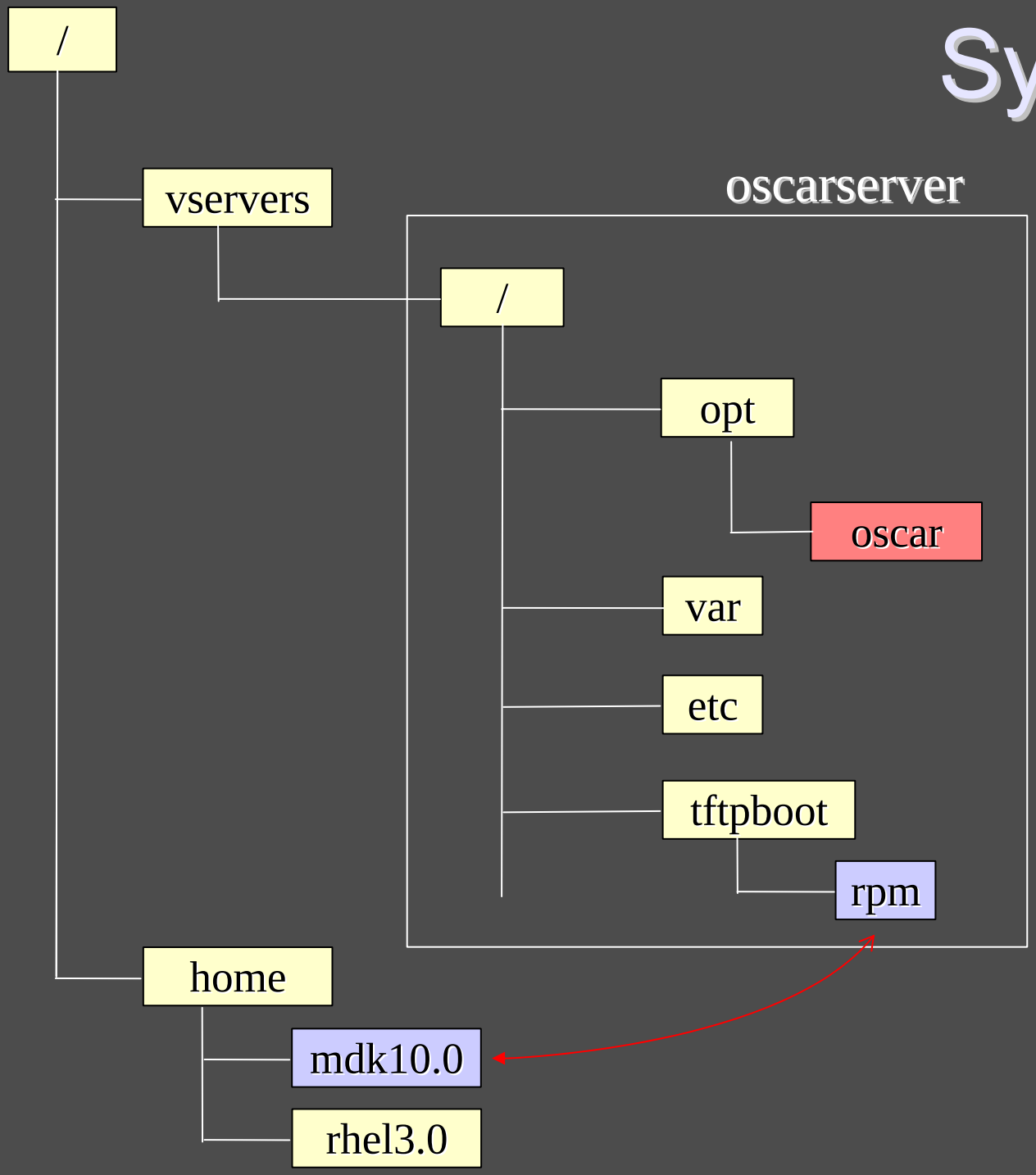
System overview

Files organization



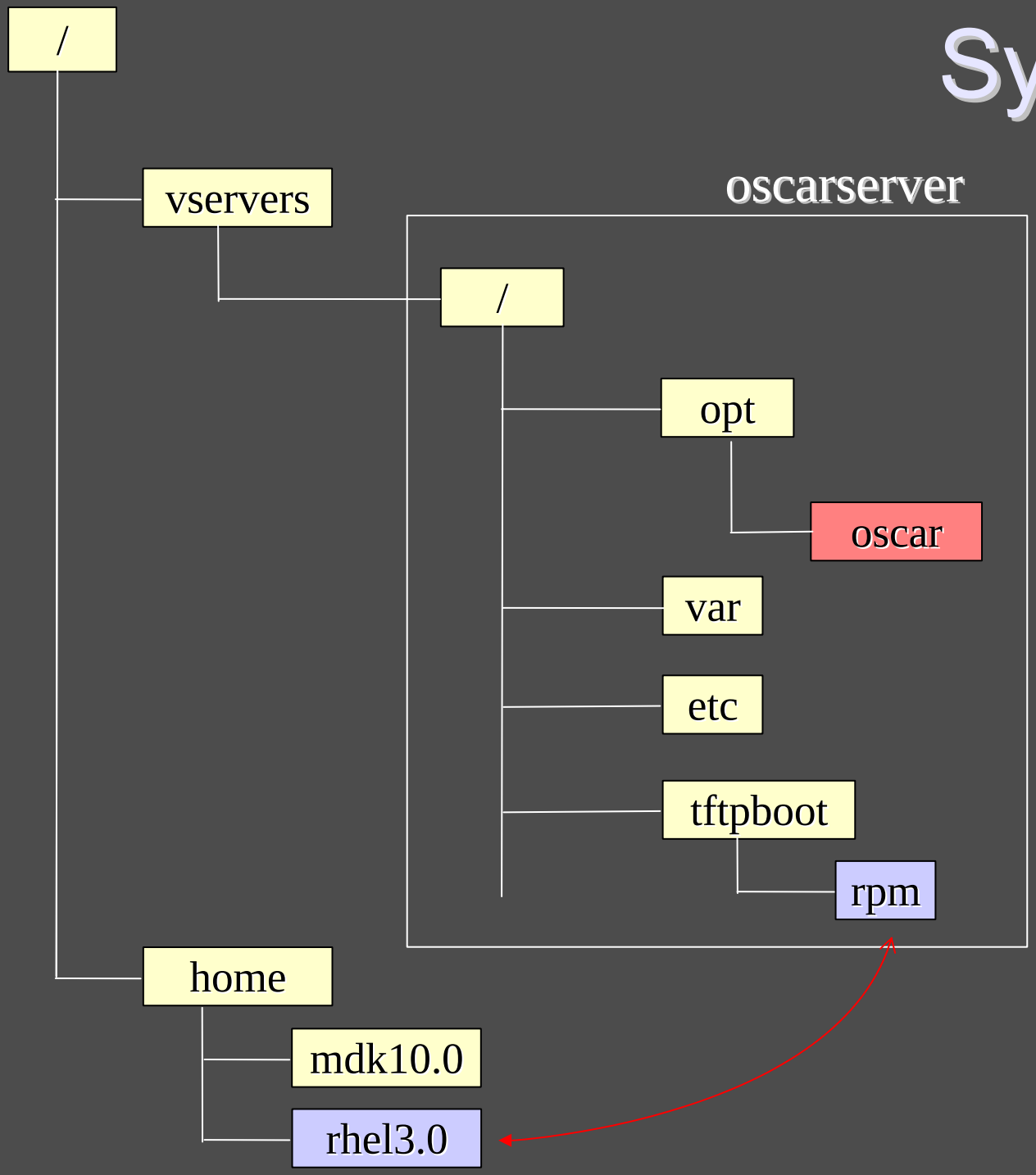
System overview

Files organization



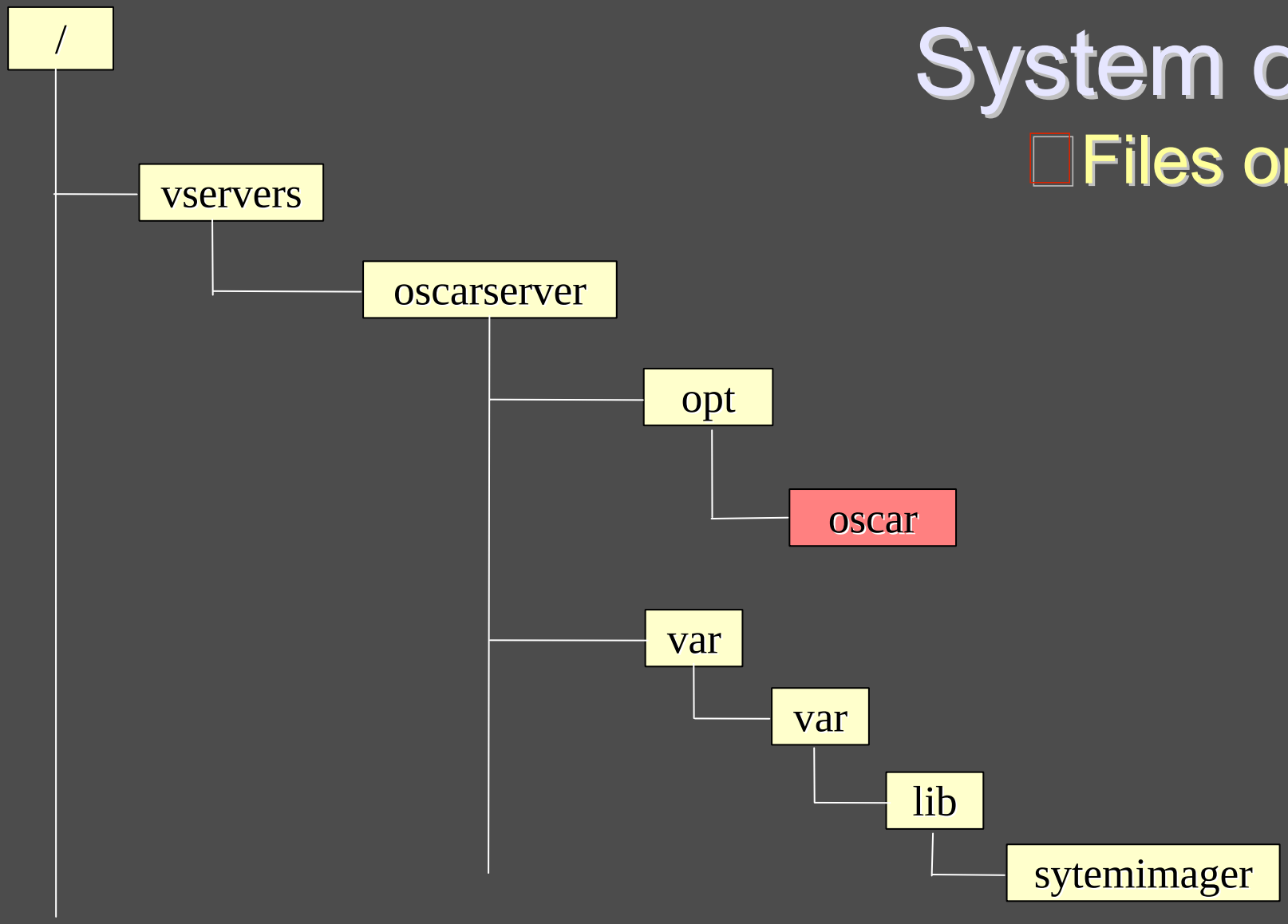
System overview

Files organization



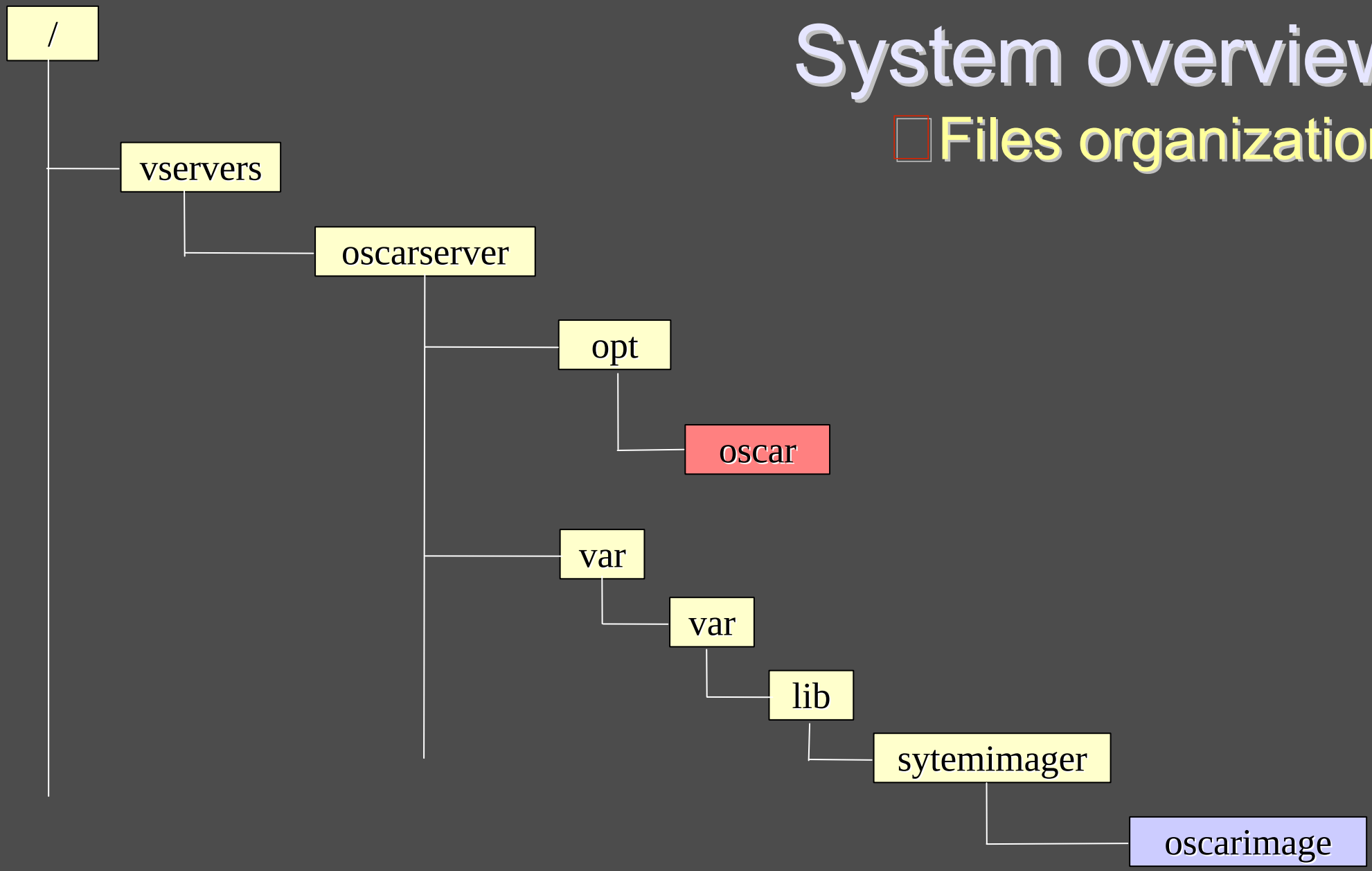
System overview

Files organization



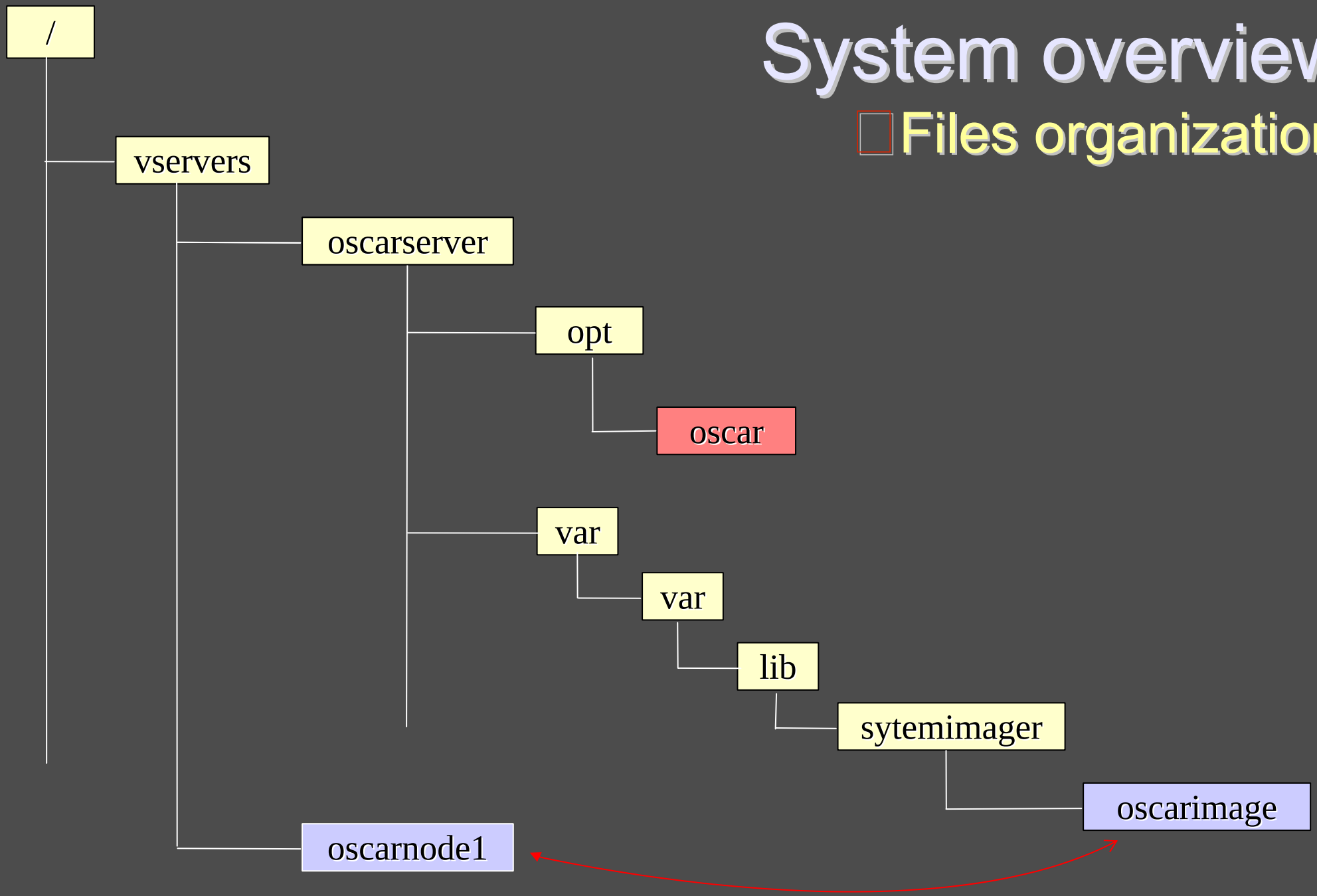
System overview

Files organization



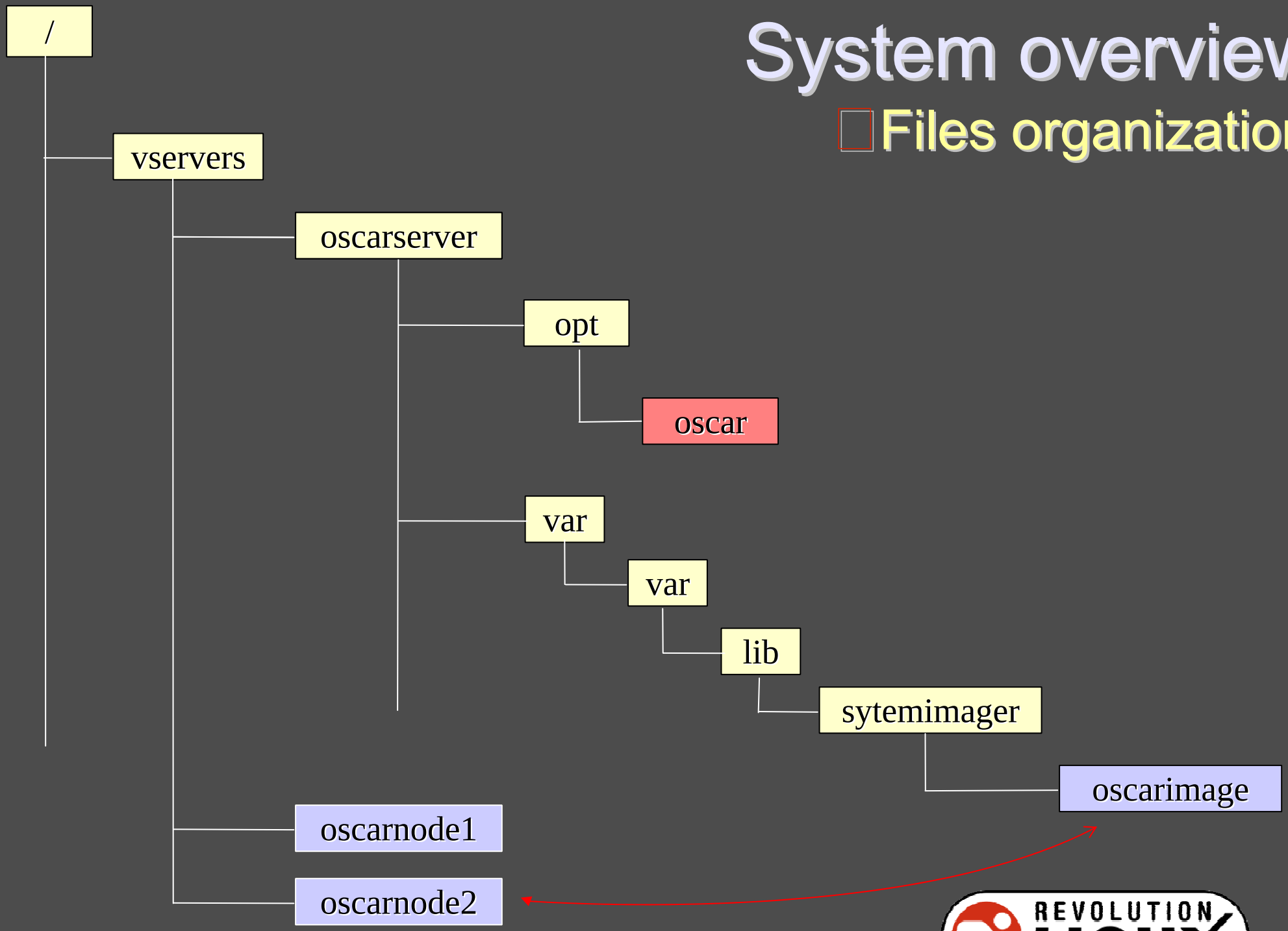
System overview

Files organization



System overview

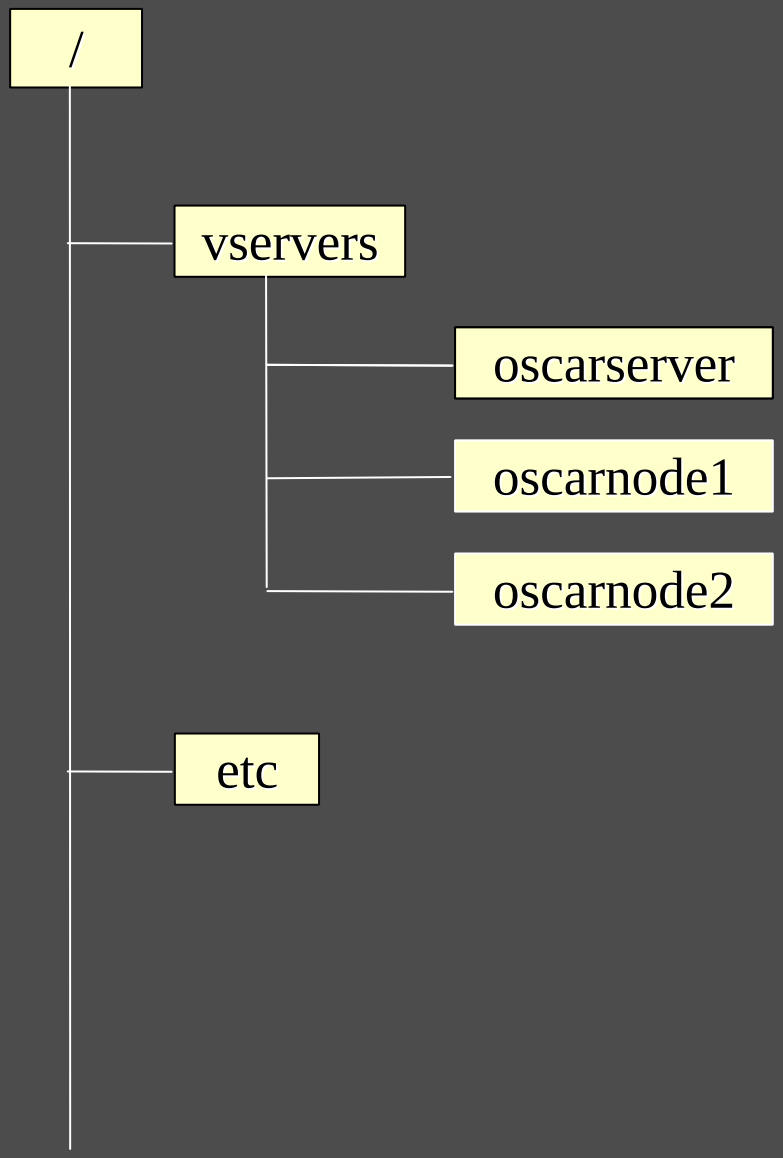
Files organization



REVOLUTION LINUX
www.revolutionlinux.com

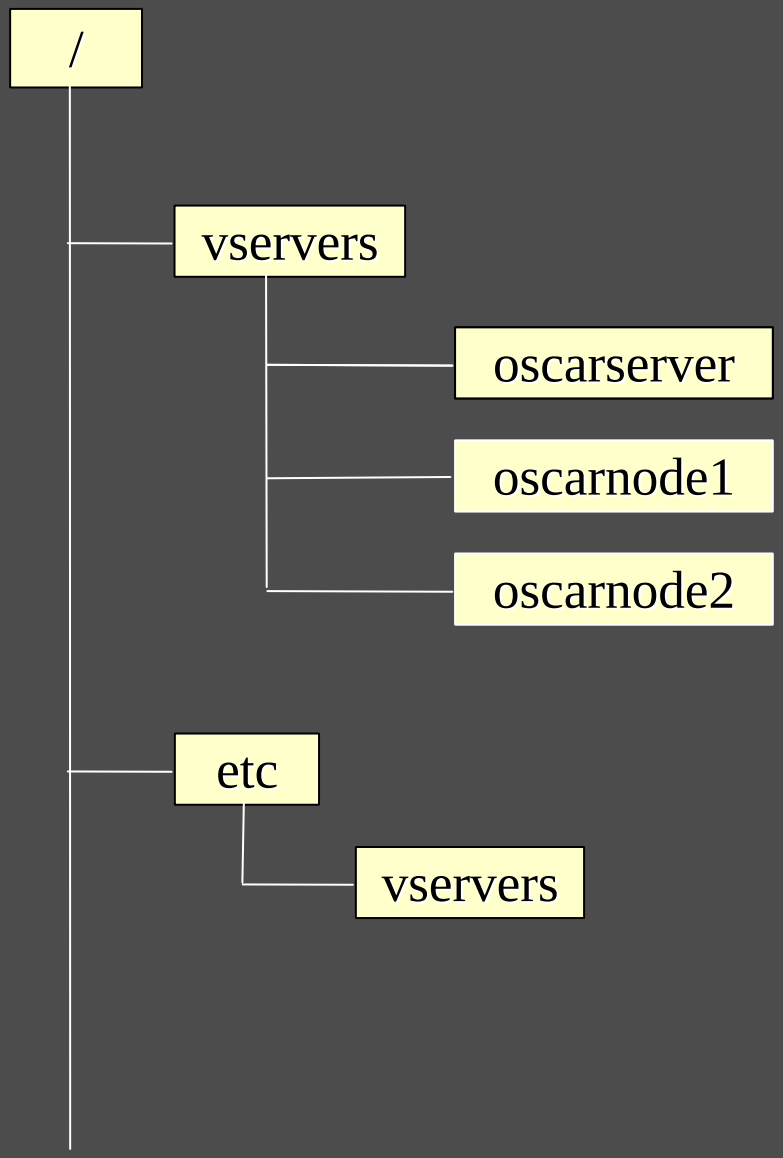
System overview

Files organization



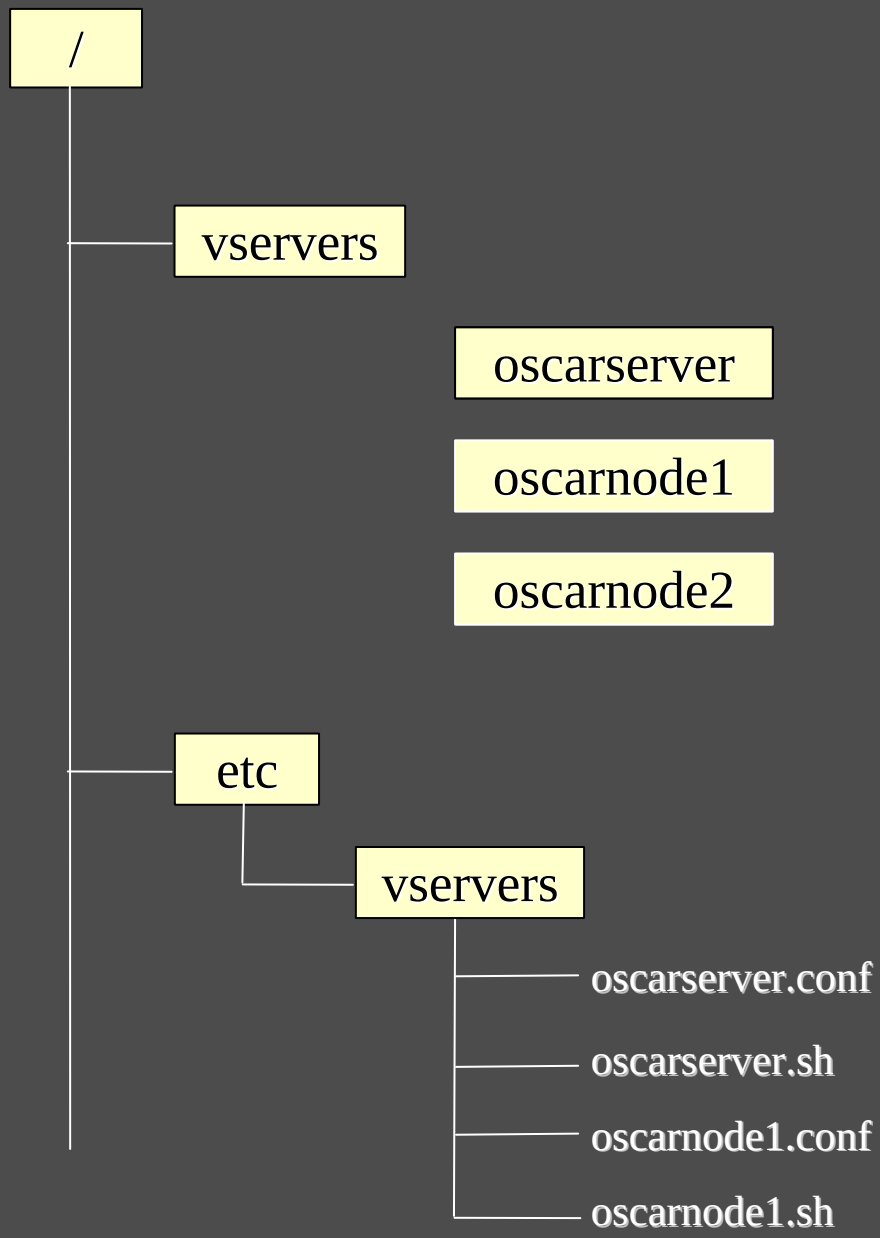
System overview

Files organization



System overview

Files organization



System overview

System overview

- From subversion to tarball

System overview

From subversion to tarball

- Once released, OSCAR is a «tarball»: *a software file containing all the necessary files and binaries to install an OSCAR cluster*

System overview

From subversion to tarball

- Once released, OSCAR is a «tarball»: *a software file containing all the necessary files and binaries to install an OSCAR cluster*
- Developers are using a subversion repository to track the changes to the source code and an automated suite of programs (autoconf, automake, make) is used to build the tarball

System overview

From subversion to tarball

- While it's possible to test directly from the subversion repository, this is not what a regular OSCAR user will use: he will use the tarball available

System overview

From subversion to tarball

- While it's possible to test directly from the subversion repository, this is not what a regular OSCAR user will use: he will use the tarball available
- So, we build the tarball from the sources, copy it to the main vserver and use it for install OSCAR

System overview

- Design overview

System overview

Design overview

- A «standard» script (script A) is used to create a vserver

System overview

Design overview

- A «standard» script (script A) is used to create a vserver
- This script is generic and can be used to create any Linux-VServer system (Fedora, RHAS, Mandrake, Debian)*

* At the time of this paper

System overview

Design overview

- A «standard» script (script A) is used to create a vserver
- This script is generic and can be used to create any Linux-VServer system (Fedora, RHAS, Mandrake, Debian)*
- This is the starting point of the automated testing process: the creation, from scratch, of a new Linux-VServer that will become the OSCAR master node

* At the time of this paper

System overview

Design overview

- Three additional scripts will be used to test OSCAR:

* At the time of this paper

System overview

Design overview

- Three additional scripts will be used to test OSCAR:

- script B: to adapt the original vserver configurations to suit OSCAR and make a copy of the subversion repository to the new vserver

* At the time of this paper

System overview

Design overview

- Three additional scripts will be used to test OSCAR:

- script B: to adapt the original vserver configurations to suit OSCAR and make a copy of the subversion repository to the new vserver

- script C: to initiate the installation process

* At the time of this paper

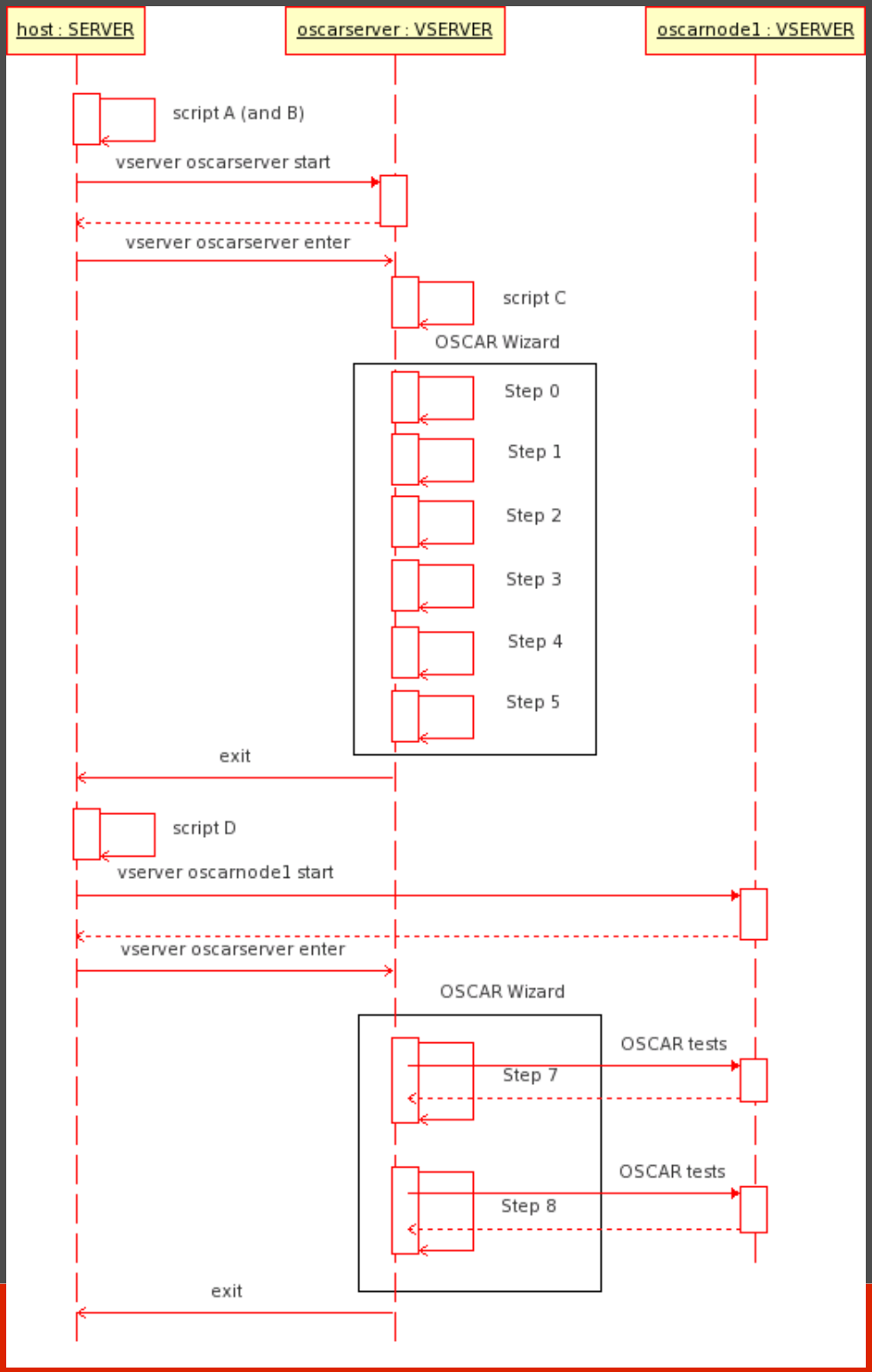
System overview

Design overview

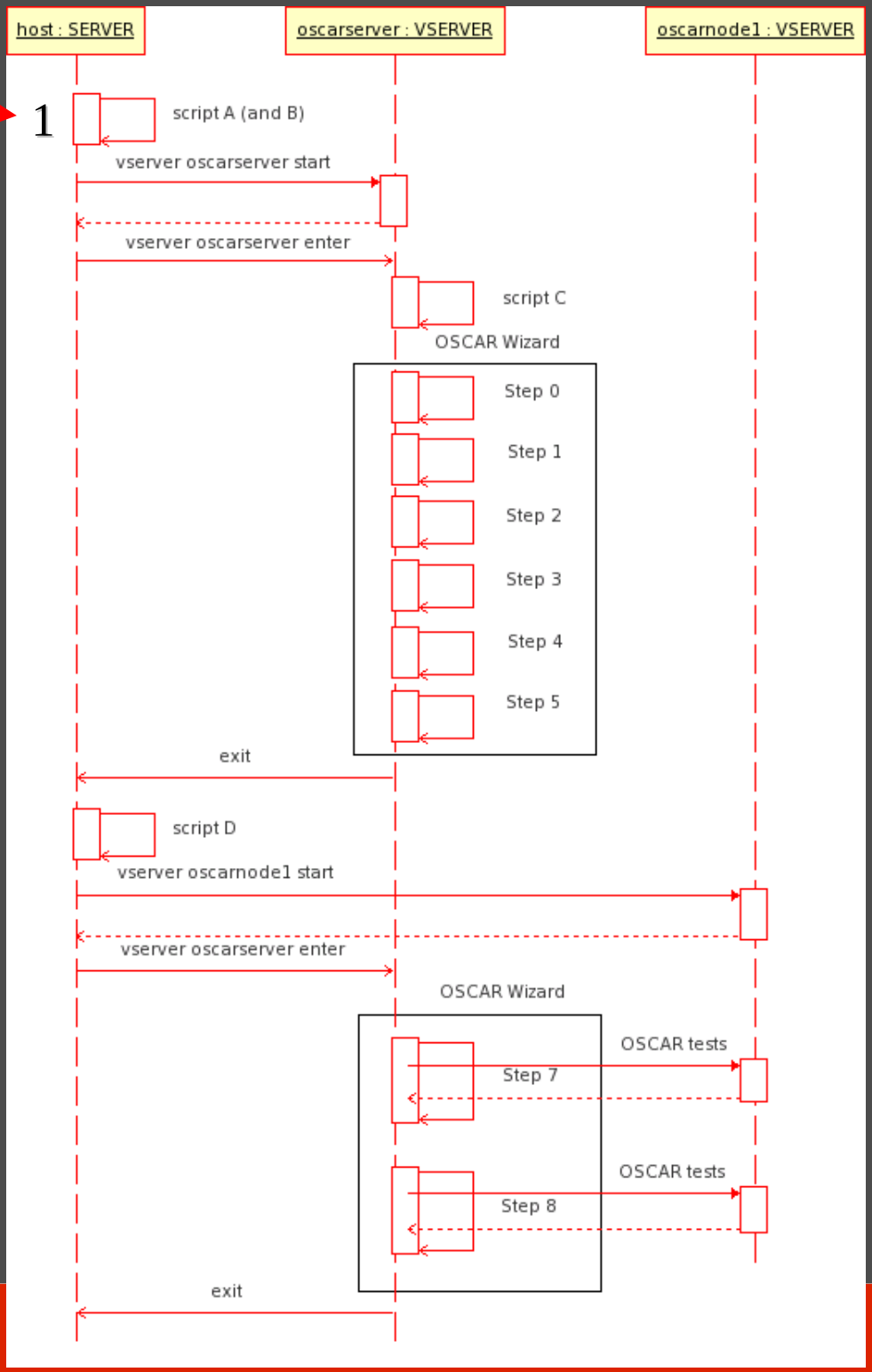
- Three additional scripts will be used to test OSCAR:
  - script B: to adapt the original vserver configurations to suit OSCAR and make a copy of the subversion repository to the new vserver
  - script C: to initiate the installation process
  - script D: to create the client nodes

* At the time of this paper

System model design

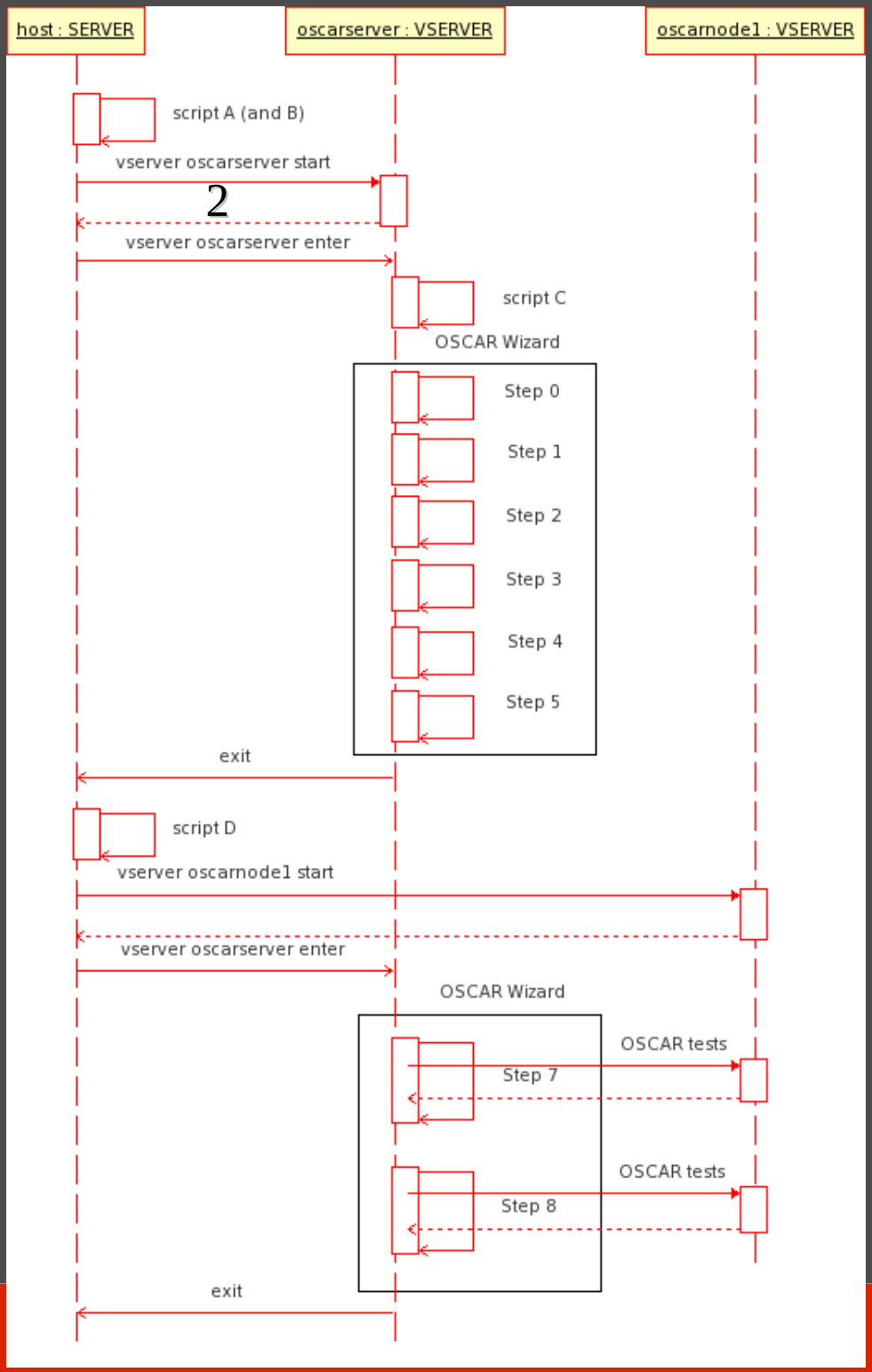


- The numbers of scripts as well as the user interaction will be reduced in the future versions in order to achieve a better automatization of the testing process



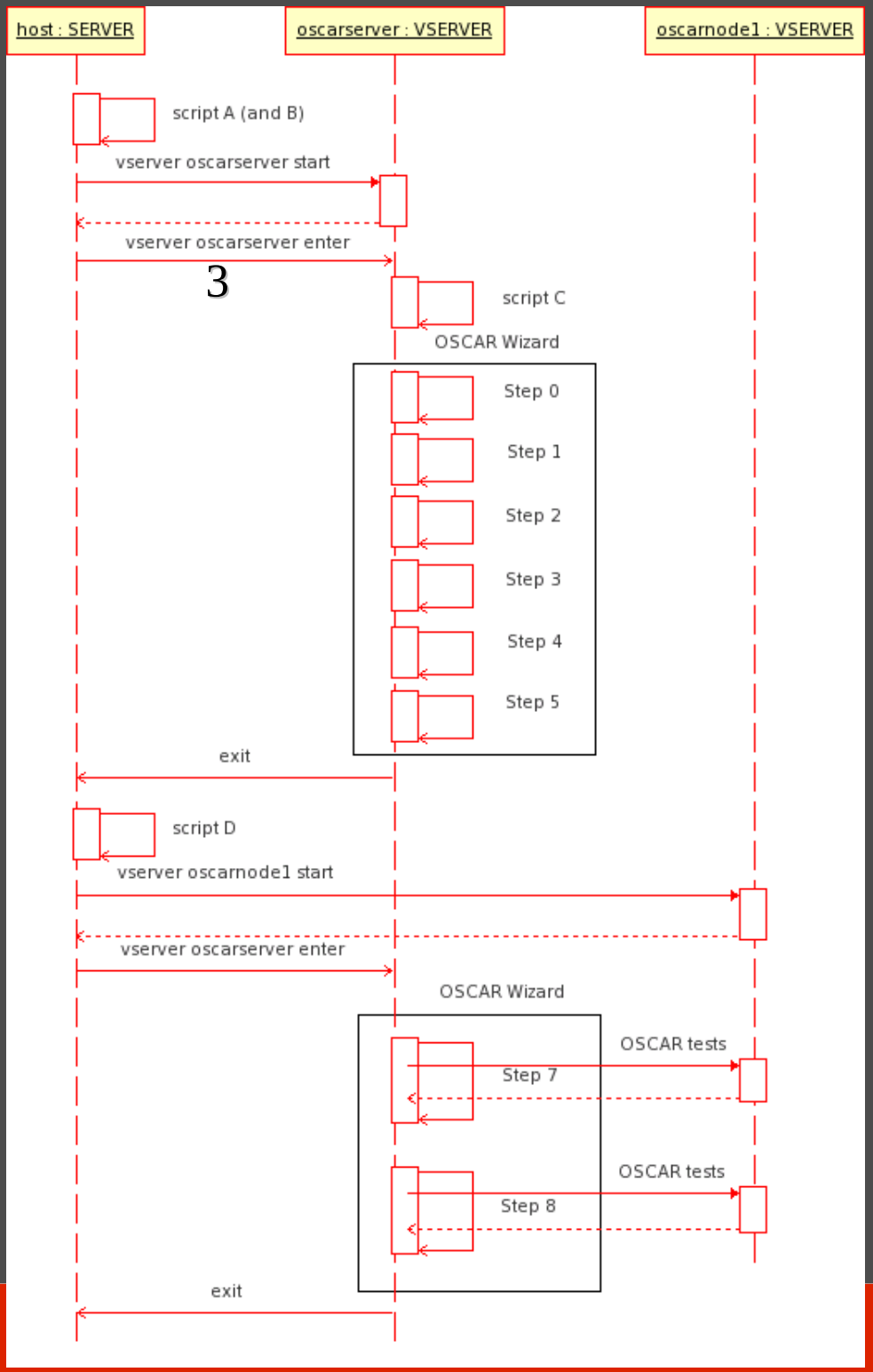
- Create the vserver (script A) with OSCAR specifics (script B)

REVOLUTION LINUX
www.revolutionlinux.com



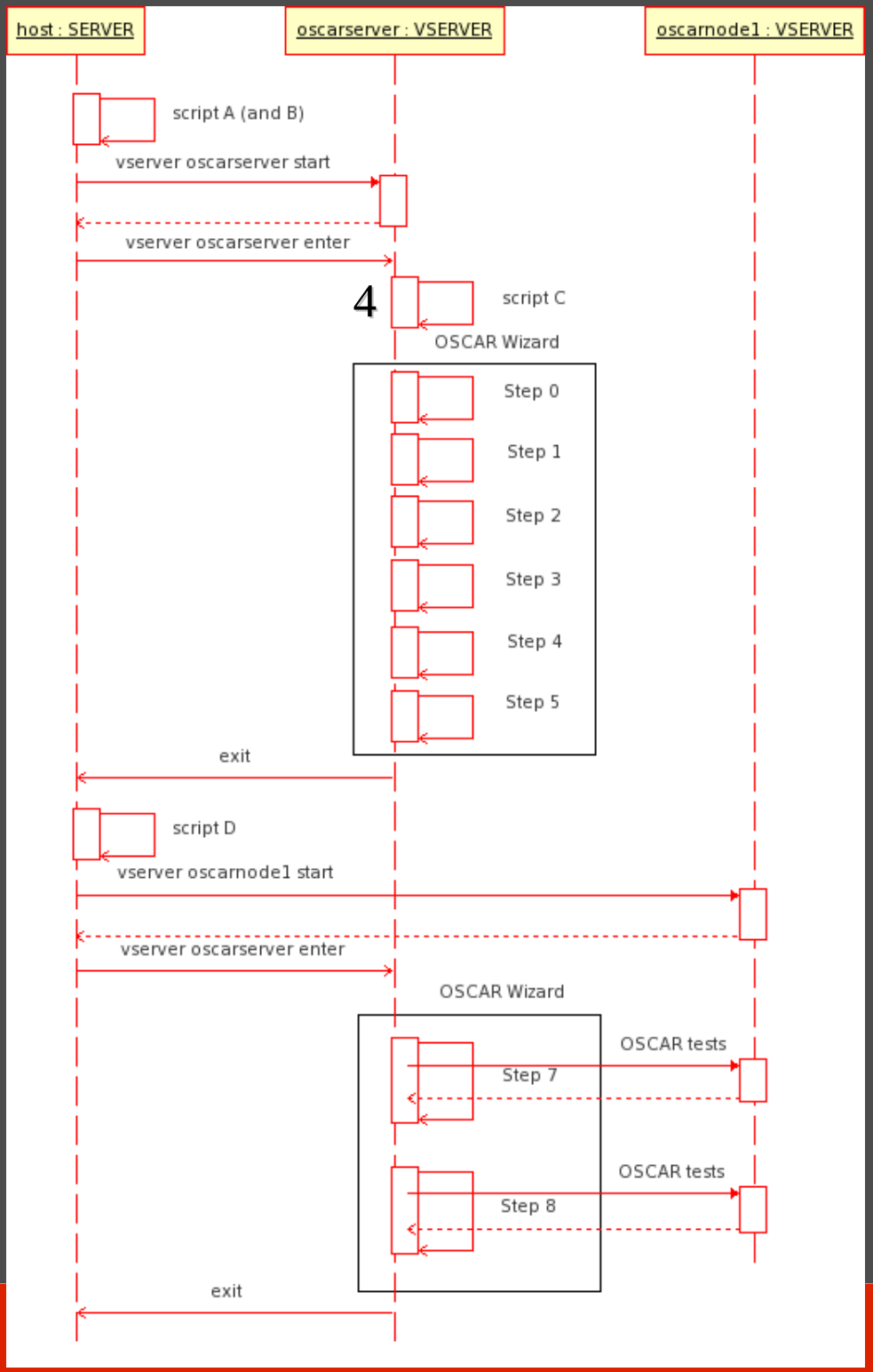
- Start the vserver master:

vserver oscarserver start



- Change to the vserver master:

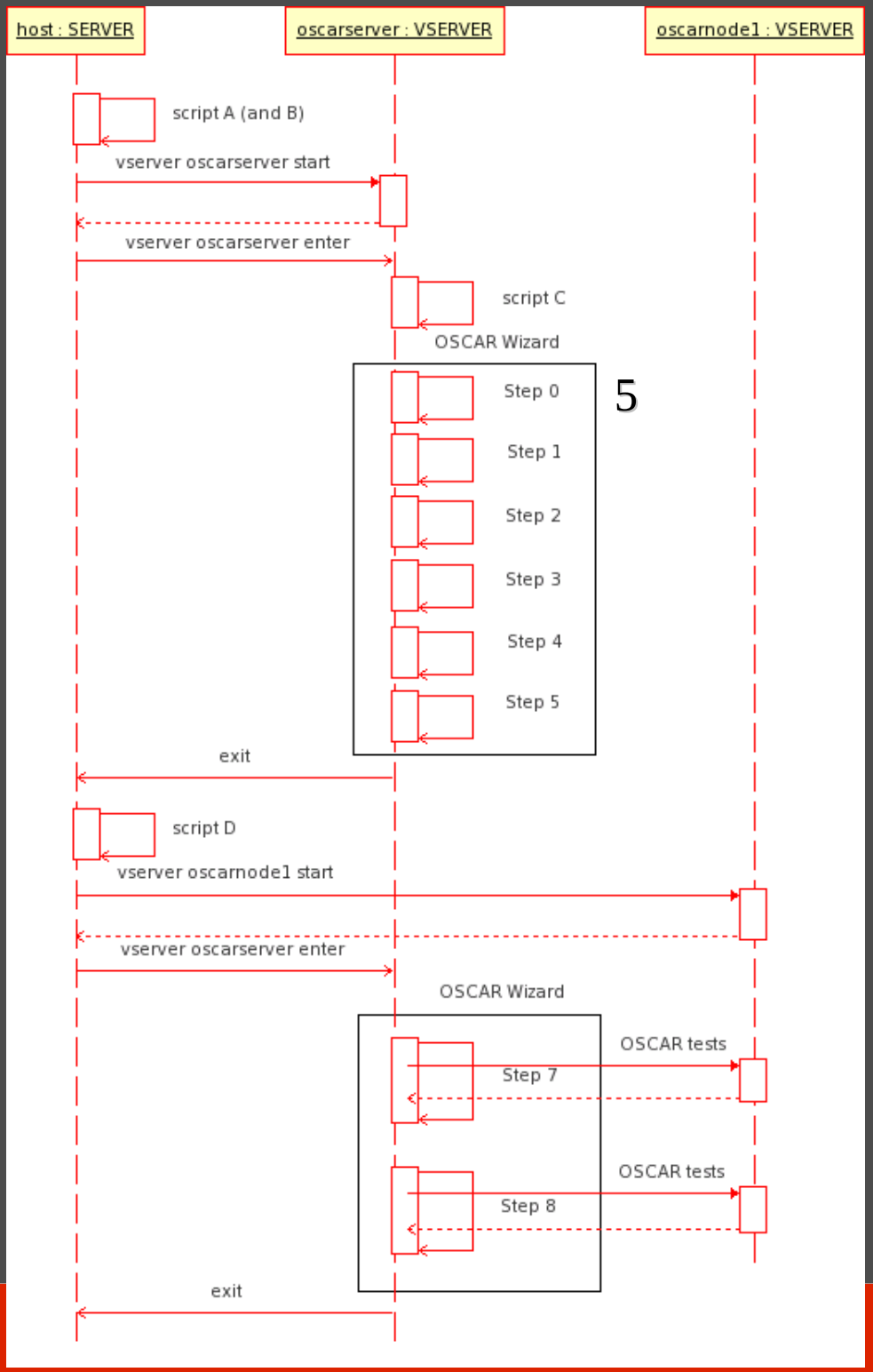
vserver oscarserver enter



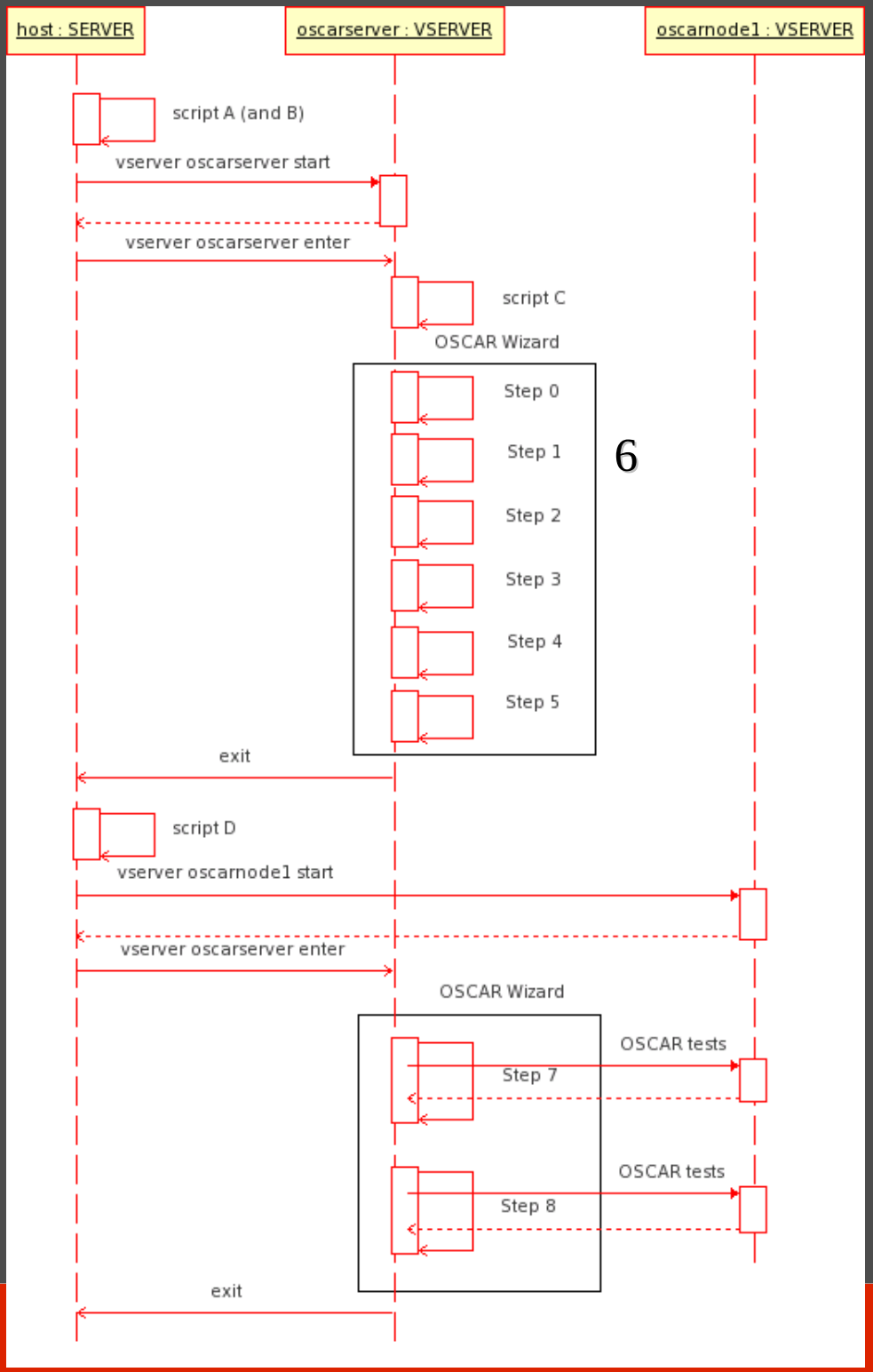
- Launch install_cluster

REVOLUTION LINUX
www.revolutionlinux.com

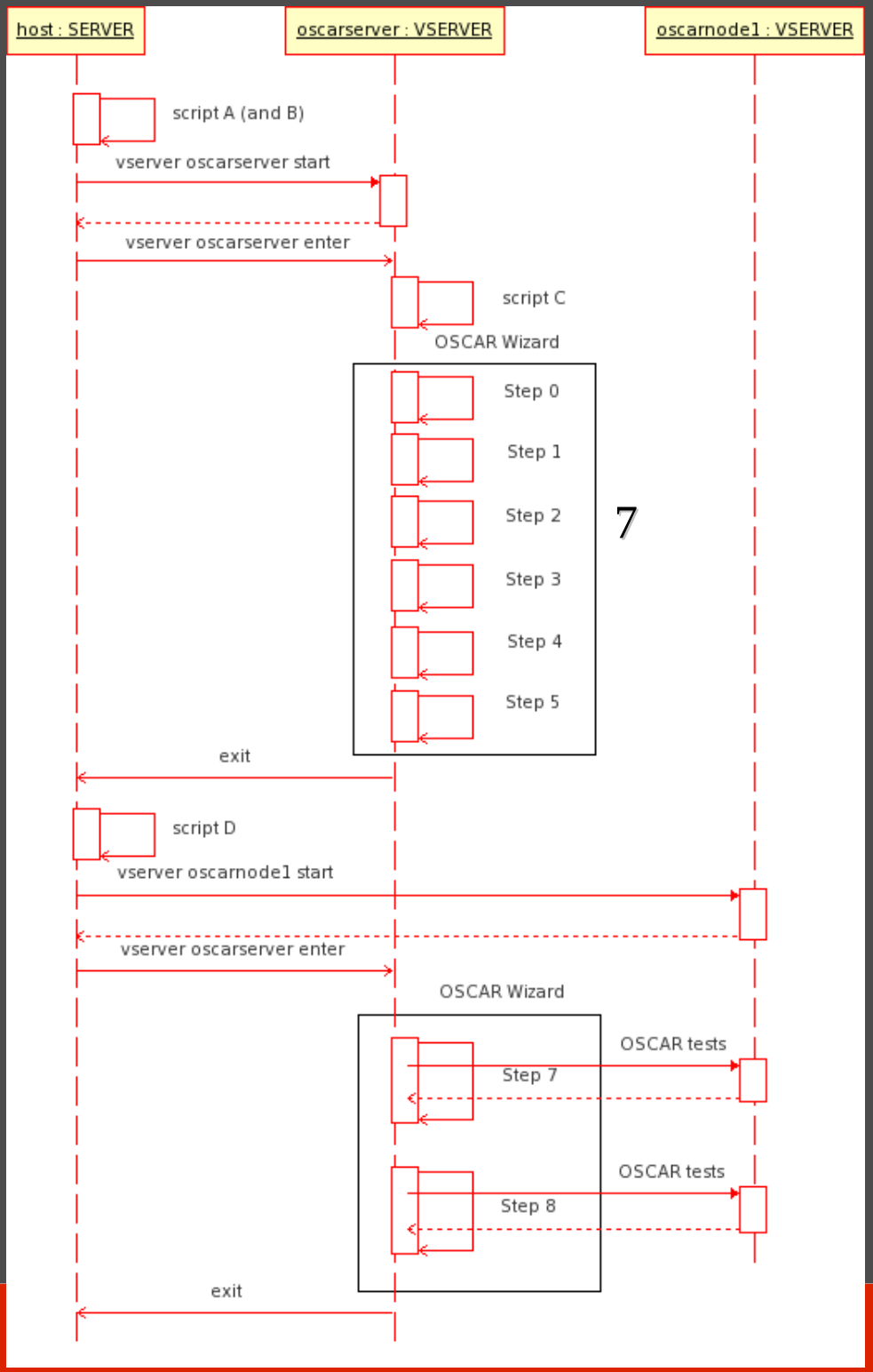
Wizard: step 0



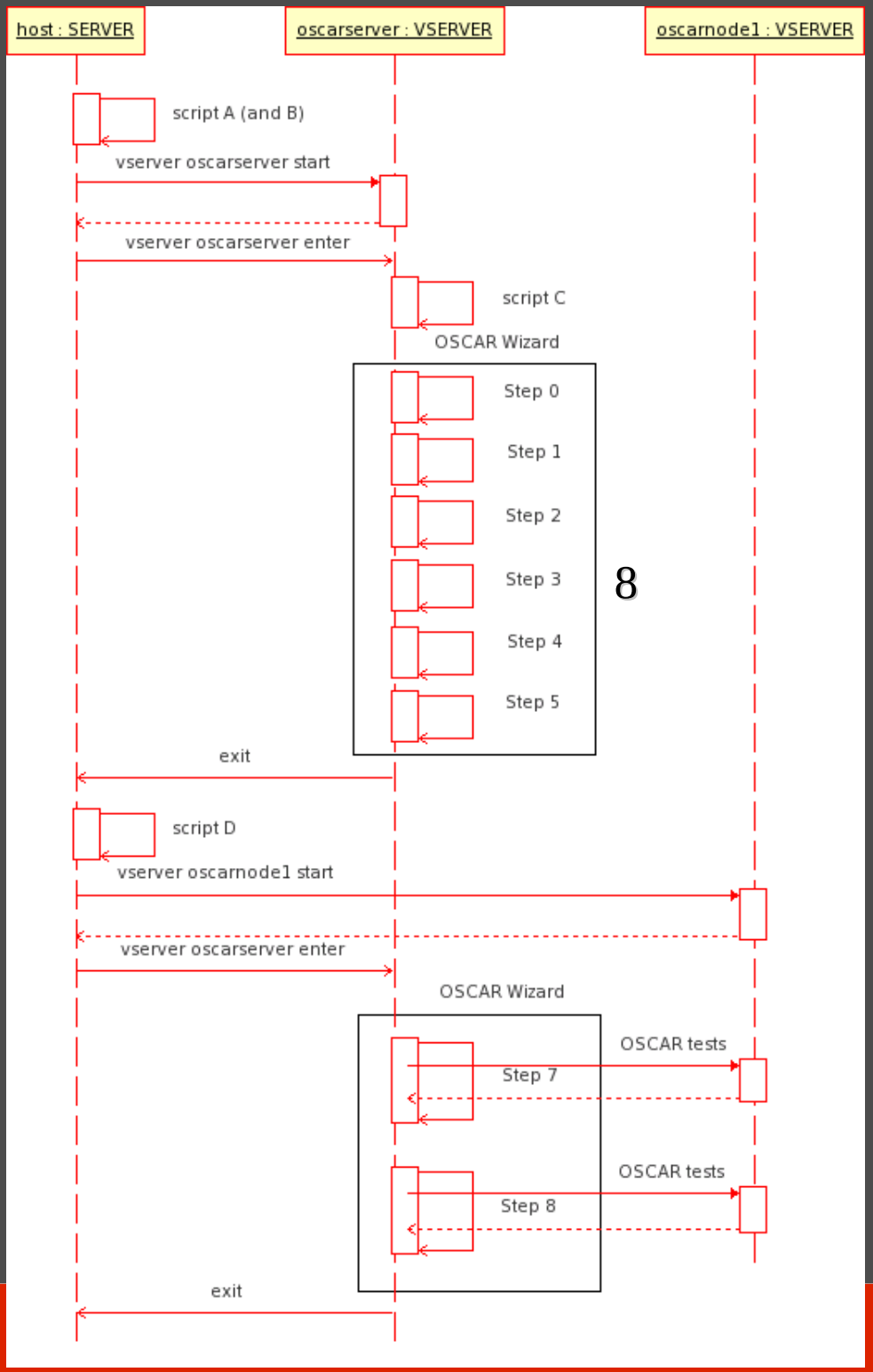
Wizard: step 1



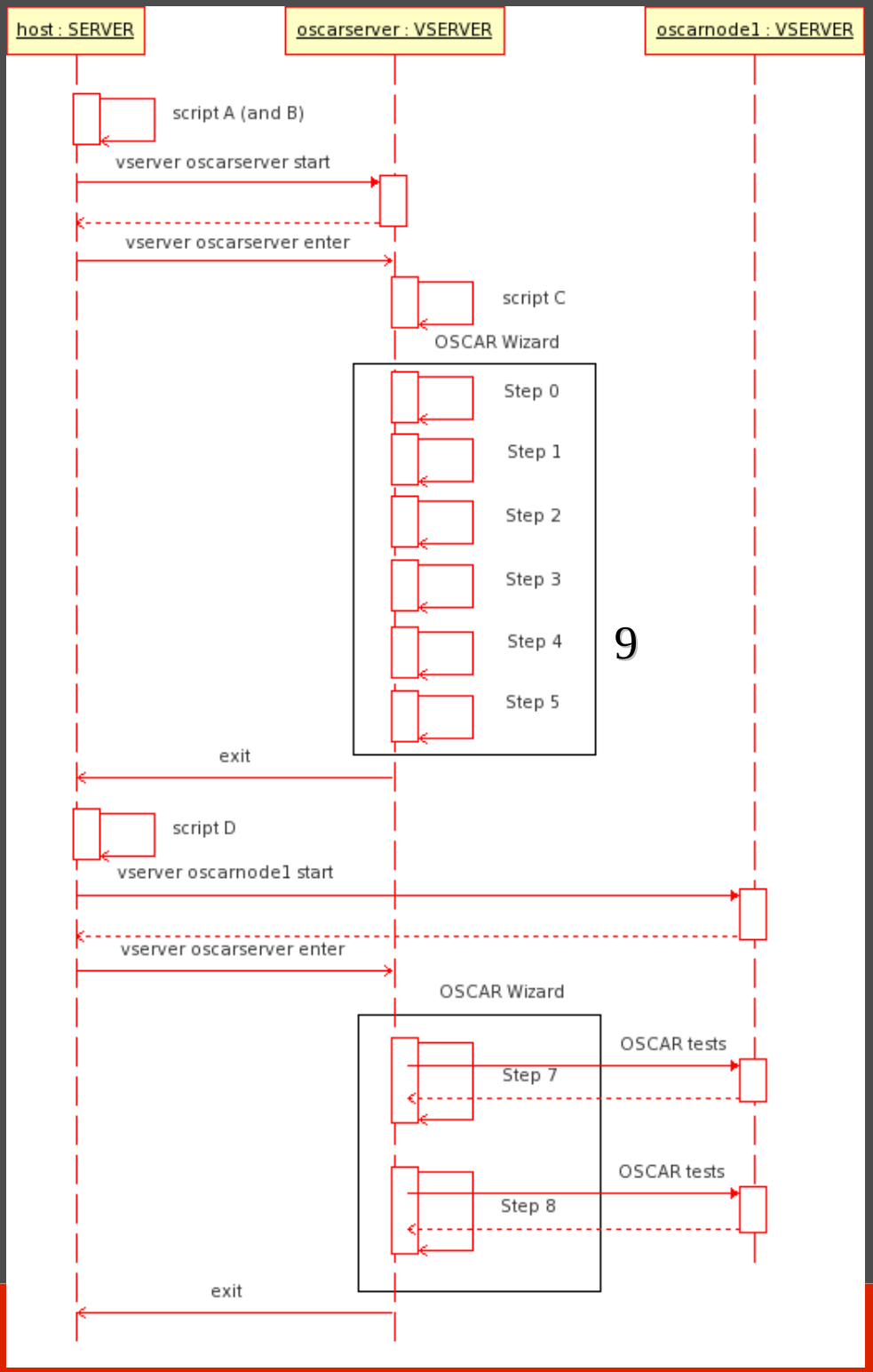
Wizard: step 2



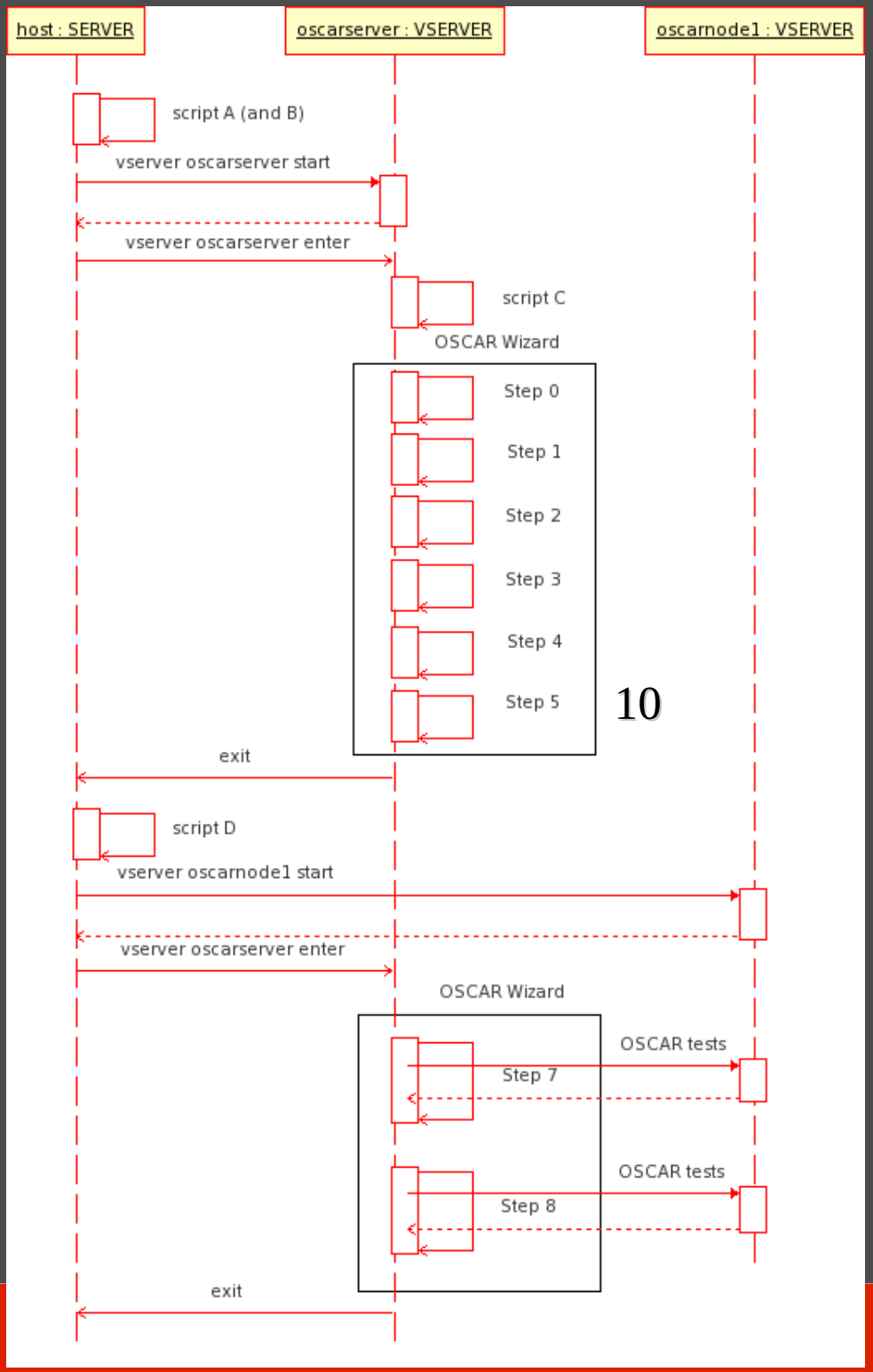
Wizard: step 3

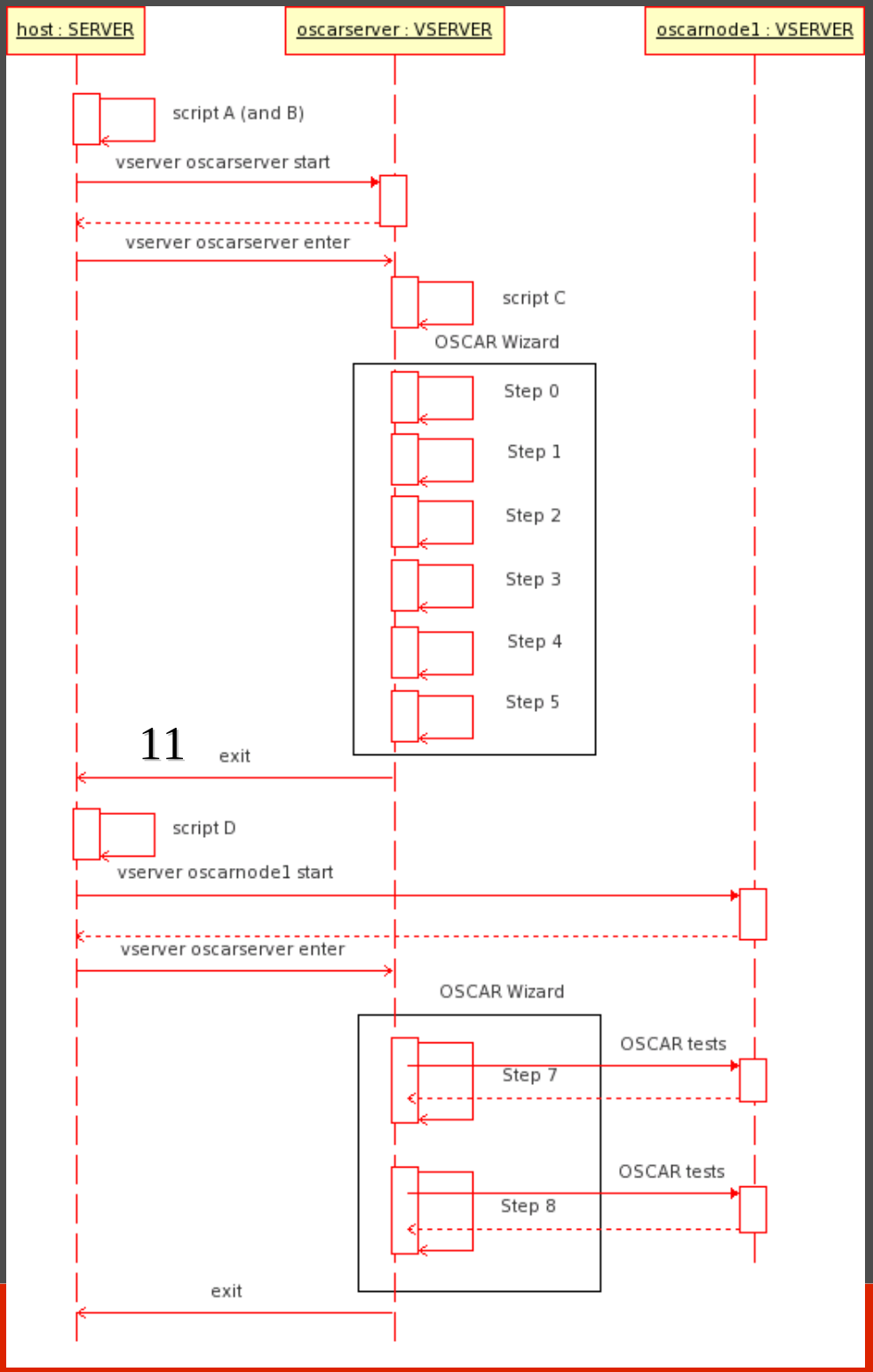


Wizard: step 4



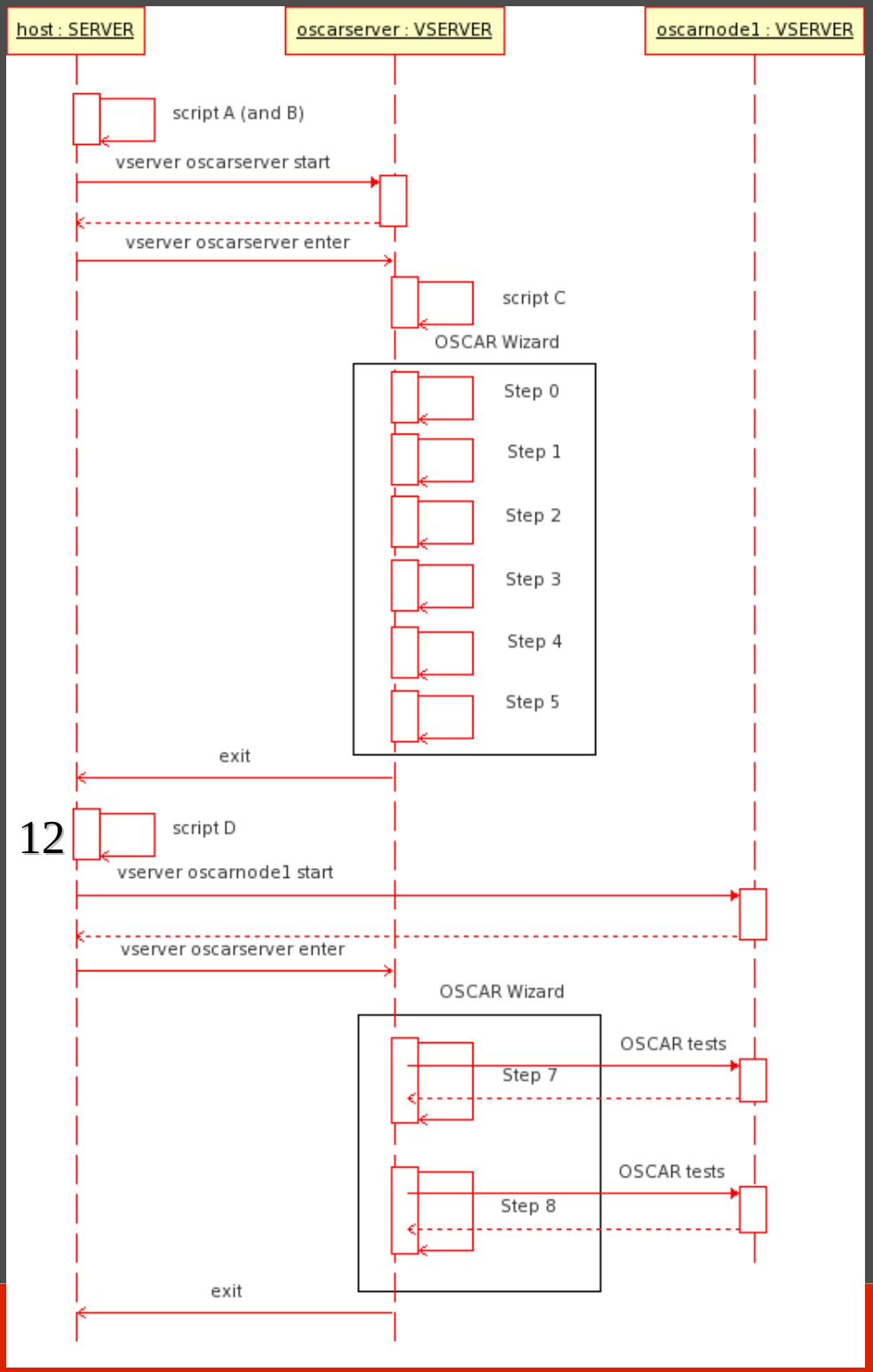
Wizard: step 5





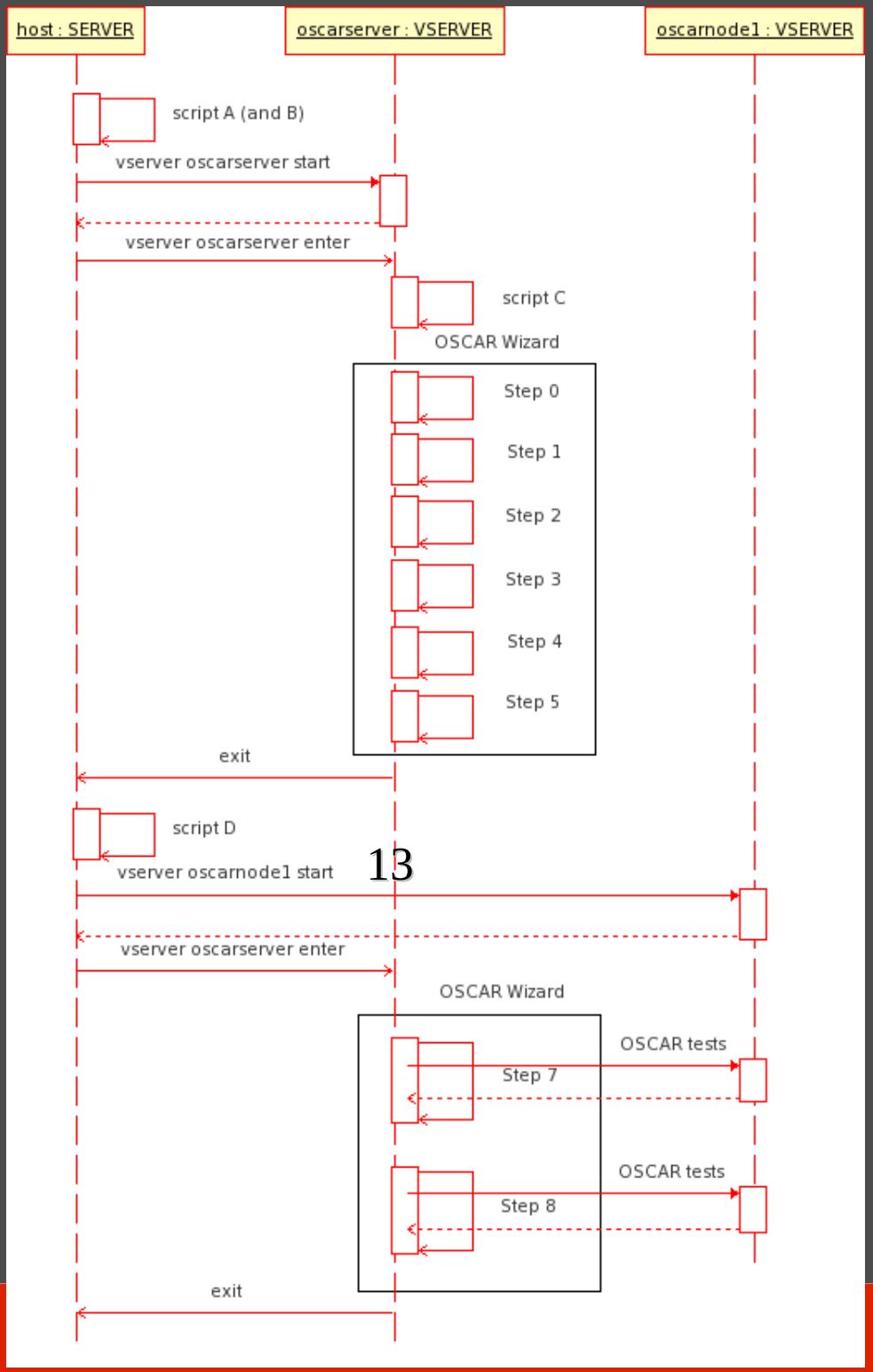
- Return to the host system

exit

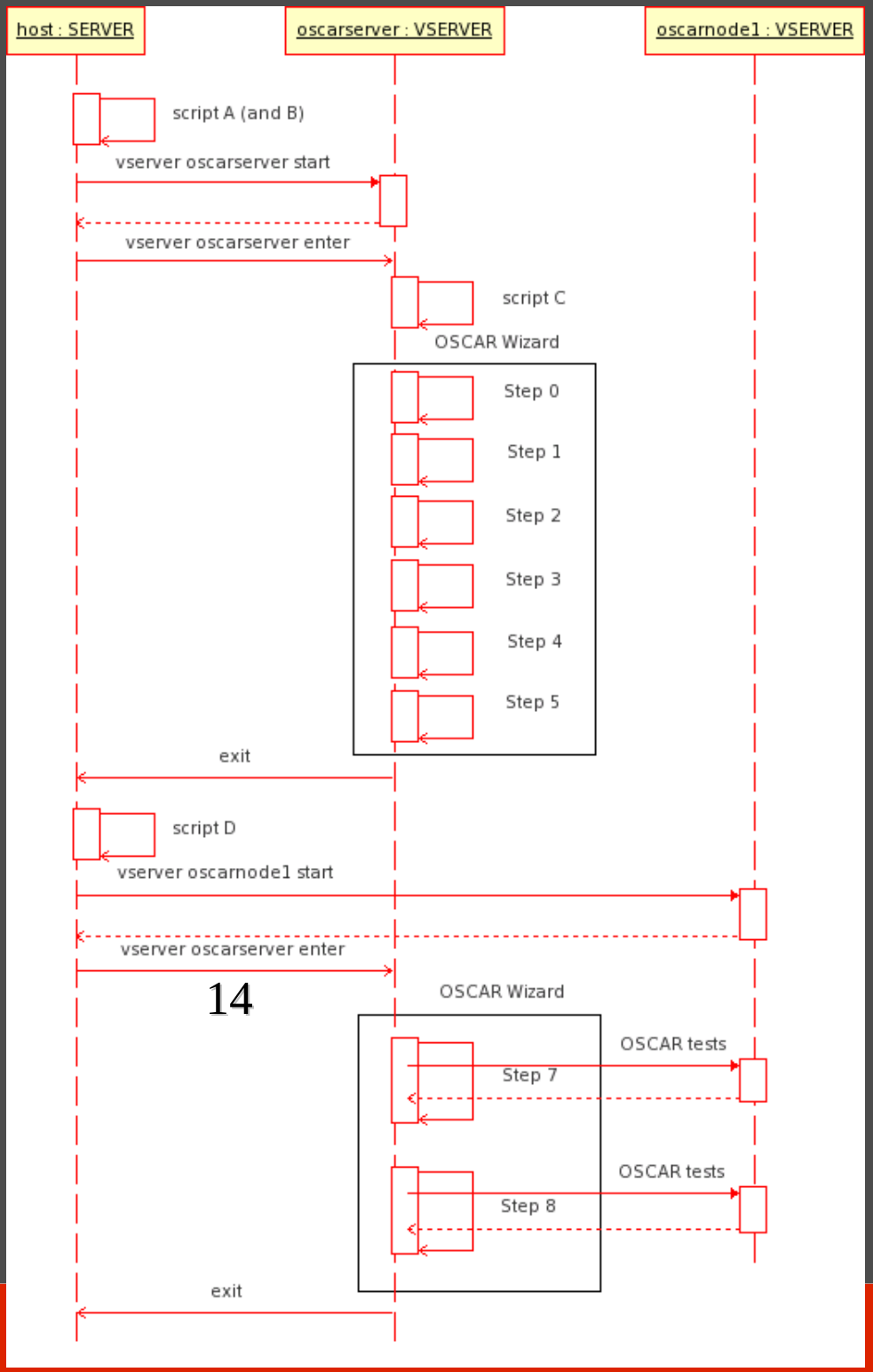


- Create the vservers clients from the oscarimage (script D)

REVOLUTION LINUX
www.revolutionlinux.com



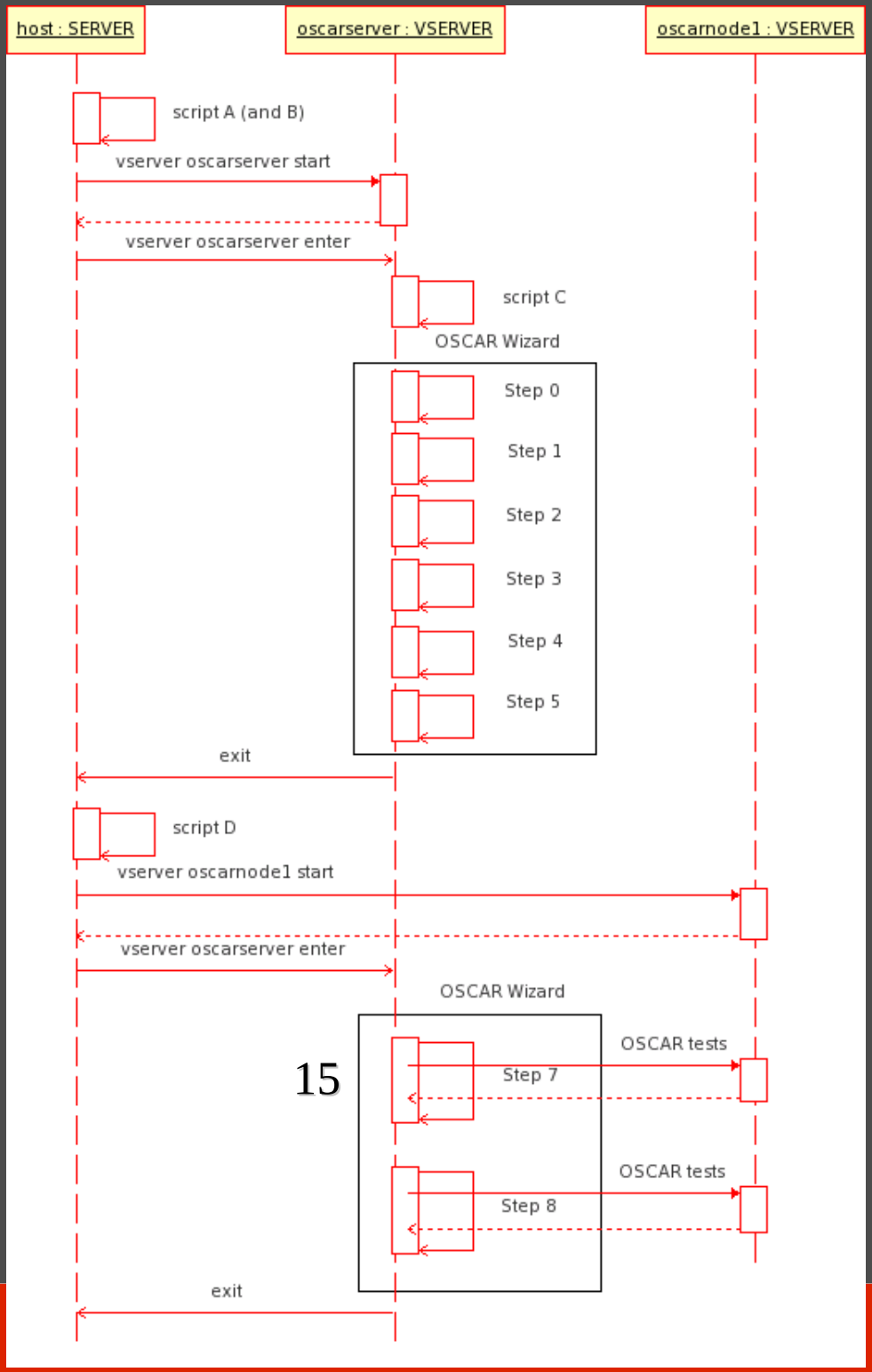
- Start each one of the clients



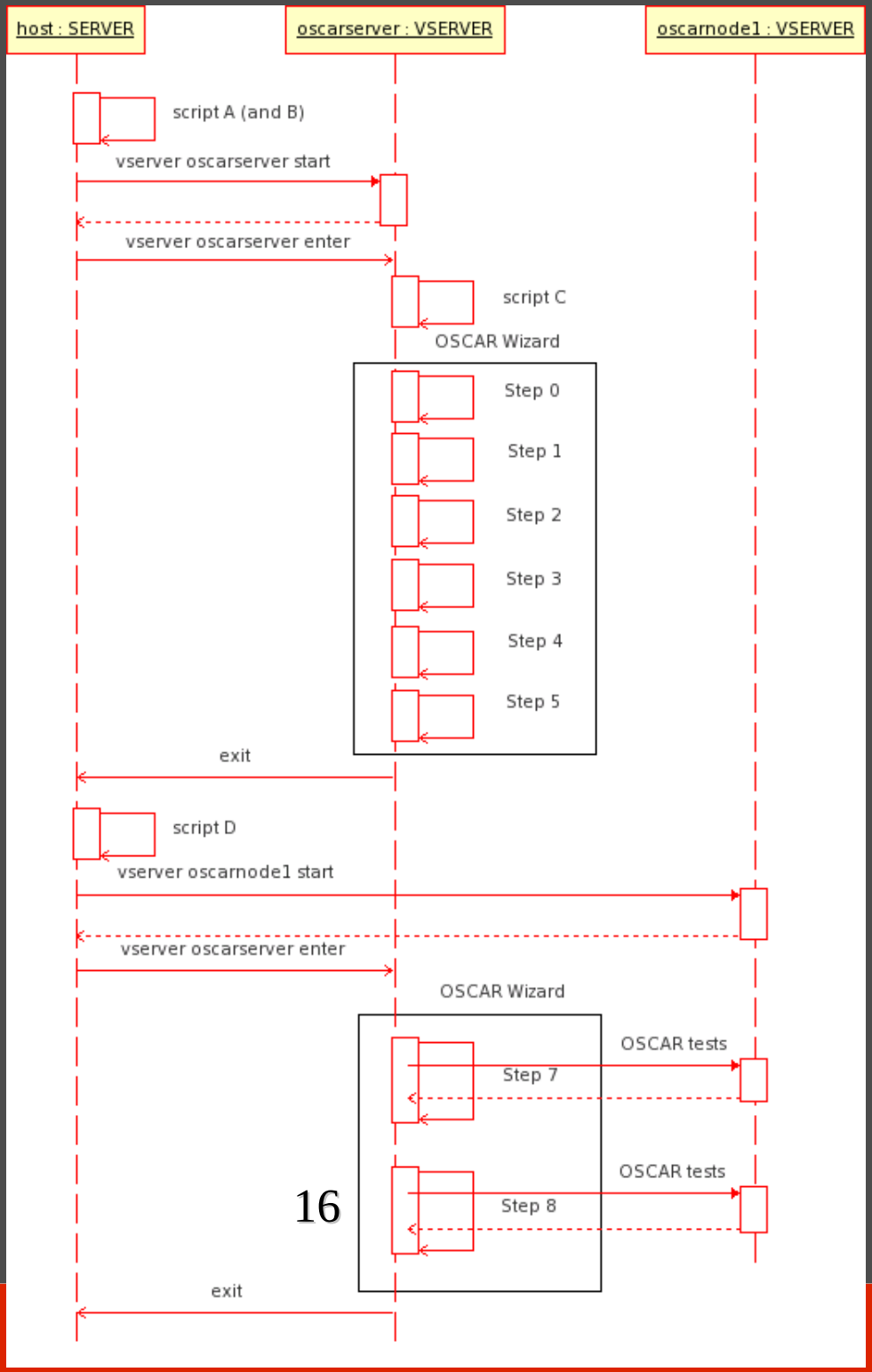
- Return to the vserver master (oscarserver)

REVOLUTION LINUX
www.revolutionlinux.com

Wizard: step 7



Wizard: step 8



System overview

- Actual implementation

System overview

Actual implementation

- Can be separated in two major steps:

System overview

Actual implementation

- Can be separated in two major steps:

- one for creating the main vserver that will host the OSCAR server (step 1)

System overview

Actual implementation

- Can be separated in two major steps:

  - one for creating the main vserver that will host the OSCAR server (step 1)

  - a second to install OSCAR (step 2)

System overview

Actual implementation

- There should not be big problems regarding the first one, once the respective RPMs are available in the default location and the system is ready to host vservers

System overview

Actual implementation

- There should not be big problems regarding the first one, once the respective RPMs are available in the default location and the system is ready to host vservers

- The testing will really begin in the second setp

Actual implementation

- Generation of the OSCAR master node

Actual implementation

Generation of the OSCAR master node

- The main «make-vserver» script (A) can be considered as a function requiring as input parameters hostname, domainname and ip address of the OSCAR server you want to generate. This «function» will return a fully functional Linux-VServer

Actual implementation

Generation of the OSCAR master node

- The main «make-vserver» script (A) can be considered as a function requiring as input parameters hostname, domainname and ip address of the OSCAR server you want to generate. This «function» will return a fully functional Linux-VServer

- The option 'TYPE=OSCAR' will trigger additional threatment of the generated vserver. This will call script B to modify the original vserver so that it can be used to build and install OSCAR (capabilities, ssh permissions, /tftpboot, etc)

Actual implementation

Generation of the OSCAR master node

- Those scripts also generate two configuration files for the vservers:

Actual implementation

Generation of the OSCAR master node

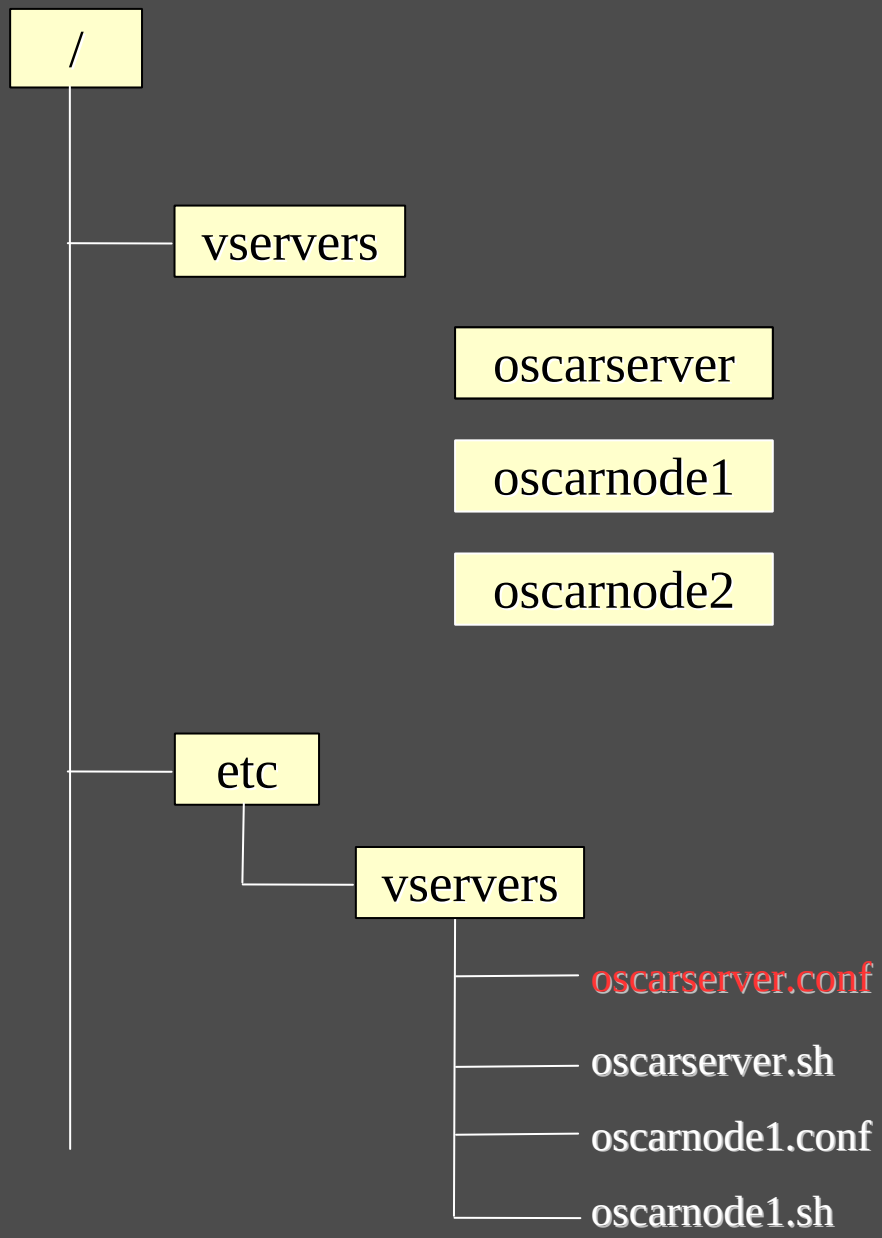
- Those scripts also generate two configuration files for the vservers:

- «vserver_name».conf: data specific to the vserver (hostname, dommainame, ip address and capabilities)

Actual implementation

Generation of the OSCAR master node

- Those scripts also generate two configuration files for the vservers:
- «vserver_name».conf: data specific to the vserver (hostname, dommainame, ip address and capabilities)
- «vserver_name».sh: contains «hooks» to scripts that should be executed pre or post-start of the Linux-VServer and pre or post-stop



oscarserver.conf

```
IPROOT=192.168.17.1
#IPROOTMASK=
#IPROOTBCAST=
IPROOTDEV=eth1
ONBOOT=yes
S_HOSTNAME=oscarserver
S_DOMAINNAME=test
S_NICE=
S_FLAGS="lock nproc"
ULIMIT="-HS -u 1000"
S_CAPS="CAP_NET_RAW CAP_MKNOD CAP_CHOWN CAP_SYS_TIME CAP_SYS_RESOURCE"
```

Actual implementation

- Generation of the OSCAR master node

Actual implementation

- Generation of the OSCAR master node

  In order to allow OSCAR to be installed inside a Linux-VServer, specific *capabilities* has to be given to this vserver:

Actual implementation

Generation of the OSCAR master node

In order to allow OSCAR to be installed inside a Linux-VServer, specific *capabilities* has to be given to this vserver:

- CAP_MKNOD: *permit the privileged aspects of mknod*

Actual implementation

Generation of the OSCAR master node

In order to allow OSCAR to be installed inside a Linux-VServer, specific *capabilities* has to be given to this vserver:

- CAP_MKNOD: *permit the privileged aspects of mknod*
- CAP_CHOWN: *change the ownership and group*

Actual implementation

Generation of the OSCAR master node

In order to allow OSCAR to be installed inside a Linux-VServer, specific *capabilities* has to be given to this vserver:

- CAP_MKNOD: *permit the privileged aspects of mknod*
- CAP_CHOWN: *change the ownership and group*
- CAP_SYS_TIME: *allow manipulation of the system clock*

Actual implementation

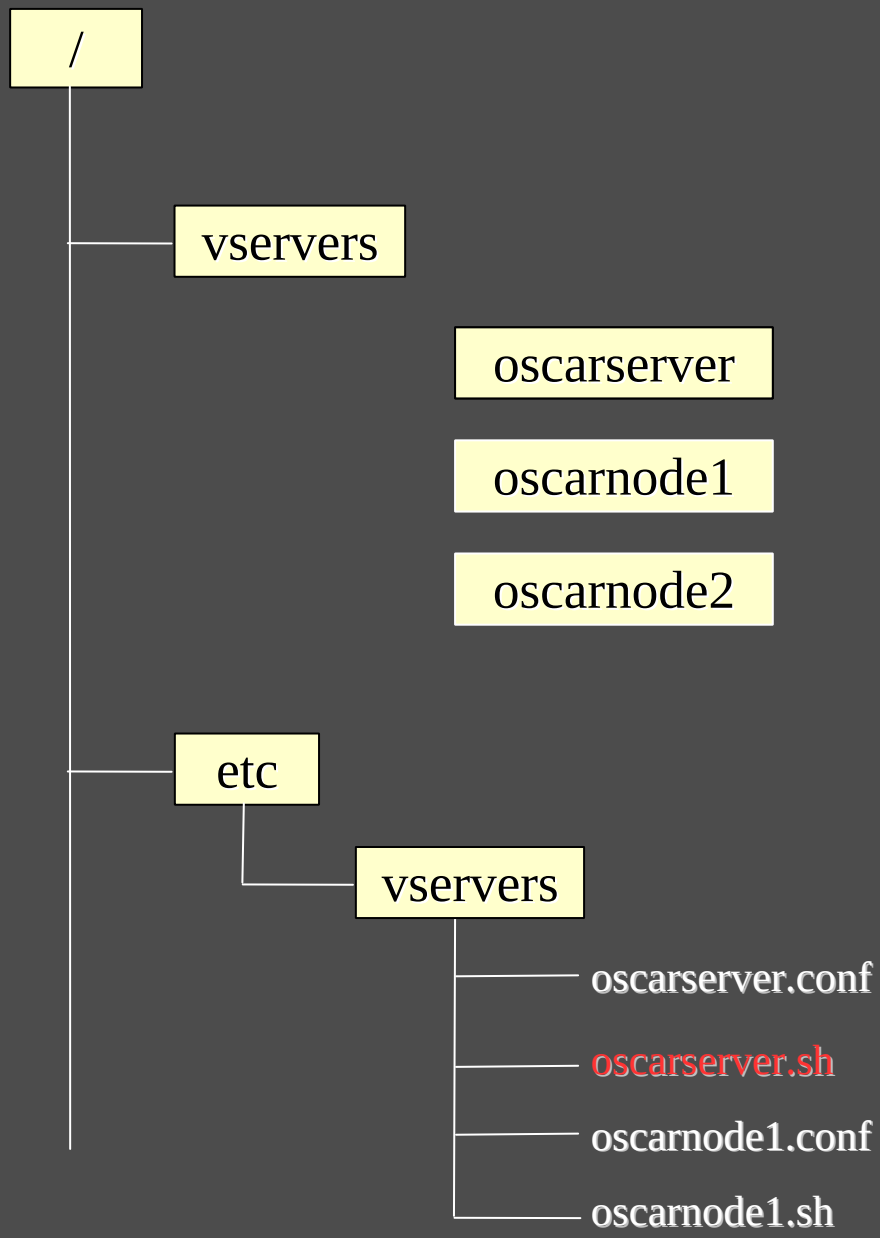
Generation of the OSCAR master node

In order to allow OSCAR to be installed inside a Linux-VServer, specific *capabilities* has to be given to this vserver:

- CAP_MKNOD: *permit the privileged aspects of mknod*
- CAP_CHOWN: *change the ownership and group*
- CAP_SYS_TIME: *allow manipulation of the system clock*
- CAP_SYS_RESOURCE: *override resource limits, quota, reserved space on fs, ...*

System overview

Files organizaton



oscarserver.sh

```
#!/bin/sh
case $1 in
pre-start)
# Ajouter les actions de pré-démarrage ici
        mount --bind /home/mdk10.0 /vservers/oscarserver/tftpboot/rpm
        vproc -e /proc/net/unix
        vproc -e /proc/net
        vproc -e /proc/net/dev
        vproc -e /dev/null
        vproc -e /proc/net/route
        vproc -e /proc/sys
        vproc -e /proc/sys/*
;;
post-start)
# Ajouter les actions de post-démarrage ici
;;
pre-stop)
# Ajouter les actions de pré-arrêt ici
;;
post-stop)
# Ajouter les actions de post-arrêt ici
        umount /vservers/oscarserver/tftpboot/rpm
;;
        *)
        echo $0 pre-start
        echo $0 pre-stop
        echo $0 post-start
        echo $0 post-stop
 ;;
esac
```

Actual implementation

Generation of the OSCAR master node

- The pre-start actions include binding the distribution RPM's repository to /tftpboot/rpm and giving to the vserve access to several entries in the /proc filesystem (using vproc)

Actual implementation

Generation of the OSCAR master node

- The pre-start actions include binding the distribution RPM's repository to /tftpboot/rpm and giving to the vserve access to several entries in the /proc filesystem (using vproc)
- Other significant and necessary change is to add localhost in the same line of the vserver ip address in /etc/hosts and delete the entry for 127.0.0.1

Actual implementation

- Installing OSCAR

Actual implementation

- Installing OSCAR

Following, some final considerations about installing oscar in the main vserver

Installing OSCAR

Installing OSCAR

- The Linux-VServer use a virtual network interface name eth*x*:«vserver_name». So, the command to launch the installation should be install_cluster eth*x*: *«vserver_name»*

Installing OSCAR

- Things must be in a well know state so that the OSCAR wizard can be loaded. Otherwise, errors will be triggered and the installation will fail, as it would in a regular environment

Installing OSCAR

- Things must be in a well know state so that the OSCAR wizard can be loaded. Otherwise, errors will be triggered and the installation will fail, as it would in a regular environment

- When the wizard appears, this mean we have a considerable part of the system operational for run OSCAR (all the prerequisites packages are installed, so no package dependencies problem)

Installing OSCAR

- The step 6 of the Wizard (*Setup Networking*) is not executed during the automated testing.

Installing OSCAR

- The step 6 of the Wizard (*Setup Networking*) is not executed during the automated testing.
- Instead of doing this step, the image created during step 4 will be copied by the script D and will be the root file system used by the newly created vservers that will act as regular OSCAR nodes.

Installing OSCAR

- The step 6 of the Wizard (*Setup Networking) is not executed during the automated testing*.
- Instead of doing this step, the image created during step 4 will be copied by the script D and will be the root file system used by the newly created vservers that will act as regular OSCAR nodes.
- Not executing *Setup Networking* means that we can not test, with the virtualization technique we use, the following functionalities: setup DHCP, listening of MAC addresses (*tcpdump*), TFTP server, rsync, remote installation, etc

Installing OSCAR

- *Because we are sharing the same hardware resources between the server and the clients we can not use NFS in order to mount the server's home directory in each client*

Installing OSCAR

- Steps 7 (*Complete Cluster Setup*) and 8 (*Test Cluster Setup*) be executed in order to test the functionality of the «brand new» cluster

Installing OSCAR

- Steps 7 (*Complete Cluster Setup*) and 8 (*Test Cluster Setup*) be executed in order to test the functionality of the «brand new» cluster
- In our tests, excepting for Ganglia, that rely on multicast (not yet supported in the Linux-VServer project) all the final tests had positive results

Installing OSCAR

- *Steps 7 (Complete Cluster Setup) and 8 (Test Cluster Setup) be executed in order to test the functionality of the «brand new» cluster*
- In our tests, excepting for Ganglia, that rely on multicast (not yet supported in the Linux-VServer project) all the final tests had positive results
- New tests should be added to exploit this testing structure.

Future contributions

Future contributions

- The automated testing structure for OSCAR is a project that is still in development.

Future contributions

- The automated testing structure for OSCAR is a project that is still in development.

- The main objective of this article is to present it to the OSCAR developer community, pointing out the advantages and possibilities that this system potentially has.

Future contributions

- There is certainly space for further development and improvements, both in its concept and actual implementation

Future contributions

- There is certainly space for further development and improvements, both in its concept and actual implementation
- Some of those will be presented in the following section

Future contributions

- There is certainly space for further development and improvements, both in its concept and actual implementation
- Some of those will be presented in the following section
- Ultimately, we hope that more ideas will come from the feedback received from you and the automated testing results

Future contributions

- Automated testing without GUI

Future contributions

Automated testing without GUI

- Because the OSCAR Wizard is, in fact, a linear process consisting of 8 steps that have to be followed in order, we think that a very simple modification to the actual code base would allow an automated testing, without wizard.

Future contributions

- Automated testing without GUI
  - Before each step, we can look at some specific place inside the /opt/oscar/automated_testing directory and search for files named accordingly to the steps: step1.xml, step2.xml, ... , step3.xml

Future contributions

Automated testing without GUI

- Before each step, we can look at some specific place inside the /opt/oscar/automated_testing directory and search for files named accordingly to the steps: step1.xml, step2.xml, ... , step3.xml

- If a file is found, then the answer to the GUI step is read from this file instead of being obtained from the GUI

Future contributions

- Automated testing without GUI

  - With this system, we can very easily implement a fully automated test plans that do not need the GUI while being very close to the actual GUI implementation

Future contributions

- Multiple distributions

Future contributions

Multiple distributions

- The structure presented here was validated using the Mandrake Linux distribution, version 10.0, and experimentally with RedHat Advanced Server 3.0

Future contributions

Multiple distributions

- It can easily be extended to cover any Linux distribution supported by OSCAR, needing only some modifications to the vservers construction scripts as well as the addition of specific rpmlists for each new distribution added

Future contributions

- Reproducing real node installation (SIS testing)

Future contributions

- Reproducing real node installation (SIS testing)

  - With the objective of having a testing structure similar to the real one where clusters are installed and used, the Linux-VServers can be used only for simulate the installation of the OSCAR server.

Future contributions

- Reproducing real node installation (SIS testing)

  - If needed, it is possible to use separate physical machine(s) as OSCAR client nodes

Future contributions

- Reproducing real node installation (SIS testing)

  - If needed, it is possible to use separate physical machine(s) as OSCAR client nodes

  - With this technique, one can still use most of the automated testing structure but manually test the physical installation process of the nodes

Future contributions

- Reference platform for OSCAR development

Future contributions

- Reference platform for OSCAR development
  - OSCAR is a layer that is installed on the top of an existing Linux installation

Future contributions

- Reference platform for OSCAR development

  - OSCAR is a layer that is installed on the top of an existing Linux installation

  - Because of this, it is very difficult to have a similar development platform common to every OSCAR developers: each one installs it's own server, with different options, partitioning, package selection, etc

Future contributions

- Reference platform for OSCAR development

  - With the use of a virtualization technique it is possible to share a so called «reference platform» for every supported architecture and Linux distribution version

Future contributions

- Reference platform for OSCAR development

  - With the use of a virtualization technique it is possible to share a so called «reference platform» for every supported architecture and Linux distribution version
  - We believe that, with this setup, bug reproduction will be much better performed

Future contributions

- Nightly testing and publication of results

Future contributions

- Nightly testing and publication of results

  - Once the tests are fully automatized, we intend to test OSCAR functionality with the latest subversion version.

Future contributions

- Nightly testing and publication of results

- Once the tests are fully automatized, we intend to test OSCAR functionality with the latest subversion version.

- Those tests will be made against a so called «OSCAR nightly build»

REVOLUTION LINUX
www.revolutionlinux.com

Future contributions

- Nightly testing and publication of results

  - We think this could greatly reduce OSCAR development time as bugs will appear very rapidly and the faulty code could be easily removed or fixed from the OSCAR subversion repository

Conclusion

Conclusion

- We presented a new and «automated» testing infrastrucutre using the virtualization technique from the Linux-VServer project.

Conclusion

- We presented a new and «automated» testing infrastrucutre using the virtualization technique from the Linux-VServer project.

- The main objective of the proposed testing infrastructure is to provide to the OSCAR developers the possibility of quickly test the development versions of OSCAR for several sets of Linux distributions/ versions/architectures without the pain and time consumption of effectively building a real cluster for this purpose.

Conclusion

- ***Due to the use of a virtualization technologie, the installation of various Linux distributions from a given host system is very easy and fully automated.***

Conclusion

- Due to the use of a virtualization technologie, the installation of various Linux distributions from a given host system is very easy and fully automated.

- This, in turn, allow the simulation of an OSCAR cluster installation, for the server and nodes, as well as the functional test of multiple distributions using several architectures

Conclusion

- ***Even if this new system can not replace completely the conventional OSCAR testing, we believe that it will contribute to the development efforts of OSCAR by drastically reducing the delay between the introduction of a new functionality and bug detection.***

Acknowledgements

This research has been funded by the National Research Council's Industrial Research Assistance Program (NRC-IRAP), project number 547017


REVOLUTION
LINUX
www.revolutionlinux.com