Agentúra
Ministerstva školstva, vedy, výskumu a športu SR
pre štrukturálne fondy EÚ

Operačný program
VZDELÁVANIE

Európska únia
Európsky sociálny fond

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Príprava štúdia matematiky a informatiky na FMFI UK v anglickom jazyku

ITMS: 26140230008

dopytovo – orientovaný projekt

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov EÚ

Users and groups

- user
  - is identified by a number – UID
  - belongs to one primary group and 0 or more supplemental groups
  - special user **root** with UID 0 – unlimited access rights
- group
  - is identified by a number – GID
  - contains 0 or more users

User database

- text file `/etc/passwd`
  - username:password:uid:gid:full name:home:shell
    - username – the name used to identify the user to humans, lower-case letters, limited to 8 characters on older systems
    - password – encrypted password, * = invalid password
    - uid, gid
    - full name – the user's real name, supplementary information
    - home – the user's home directory
    - shell – the shell started when the user logs in (valid shells are specified in `/etc/shells`)

User database

- the `/etc/passwd` file has to be readable for all users in the system in order to allow them to map UIDs to usernames (e.g. in `ls`)
- newer systems use **x** in place of the password in `/etc/passwd` and store passwords in `/etc/shadow`
- `/etc/shadow` is readable only for **root**

User database

- text file /etc/shadow
  - username:password:last changed:min age:
    max age:warn before:lock after:acc. exp.:reserved
    - last changed – the date of the last change of the password (in days since 1.1.1970)
    - min. age – the min. number of days before the user can change the password
    - max. age – the max. number of days before the password must be changed
    - warn before – number of days before password expiry when the system warns the user
    - lock after – number of days after password expiry when the account is locked
    - acc. exp. - the date when the account will be locked

Group database

- text file `/etc/group`
  - groupname:password:gid:user list
  - groupname – the name of the group
  - password – encrypted password of the group (or empty)
  - gid – the group ID number
  - user list – comma-separated list of users for whom this group is a supplementary group
- text file `/etc/gshadow`
  - groupname:password::

An example of /etc/passwd

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
nobody:x:99:99:nobody:/:
janko:x:1000:100:Janko Hrasko,,,:/home/janko:/bin/bash
```

An example of /etc/shadow

```
root:nhC.YP4s8lF1Y:11783:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
mysql:*:9797:0:::::
gdm:*:9797:0:::::
nobody:*:9797:0:::::
janko:in9.jjl2XgsXQ:11783:0:99999:7:::
```

An example of /etc/group

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root,jerry
mail::12:mail
news::13:news
uucp::14:uucp,jerry
man::15:
games::20:
slocate:x:21:
mysql::27:
gdm::42:
ftp::50:
nobody::98:nobody
```

Creating a user

- by hand – by editing `/etc/passwd`, `/etc/shadow`, and `/etc/group`
- `useradd [-c FullName] [-m] username`
  - `-m` also creates home directory
  - `-g grp` - primary group for the user
  - `-G grp[,...]` - supplementary groups for the user
  - `-s shell, -e YYYY-MM-DD,  -f inact`
  - `-D [-g grp] [-b home] [-s shell] [-e expiry] [-f inact]`
- script adduser

Modifying and deleting a user

- deleting:
  `userdel [-r] username`
  - `-r` – also deletes the user's home directory
- modifying:
  `usermod options username`
  - `-c FullName, -d home [-m], -g grp, -G g1,g2,..., -l newname, -s shell, -e exipry_date, -f inact_days`
  - `-L, -U` – lock, unlock

Changing password

- change password: `passwd username`
- lock password: `passwd -l username`
- unlock password: `passwd –u username`
- other switches allow you to set expiration attributes
- other commands for modifying selected user's attributes:
  - chsh (change shell), chfn (change FullName), chage (change password expiration attributes)

Creating and deleting a group

- by hand – by editing `/etc/group`
- creating:
```
groupadd groupname
```
- deleting:
```
groupdel groupname
```
- renaming:
```
groupmod -n newname oldname
```

Processes

- every process is assigned
  - UID
    - real – UID of the user who started the process
    - effective – UID of the user whose access rights the process currently uses
    - saved
  - GID
    - real, effective, saved
  - list of supplementary group IDs

Access rights to the filesystem

- every filesystem object is assigned
  - owner – UID of a user
  - group – GID of a group
  - access rights (permissions)
    - read (r) – read from the file/directory
    - write (w) – write to the file, change the contents of the directory (create/rename/delete a directory entry)
    - execute/search directory – execute the file, use the directory in a path to a filesystem object
  - for the owner, for the group, and for others

Access rights to the filesystem

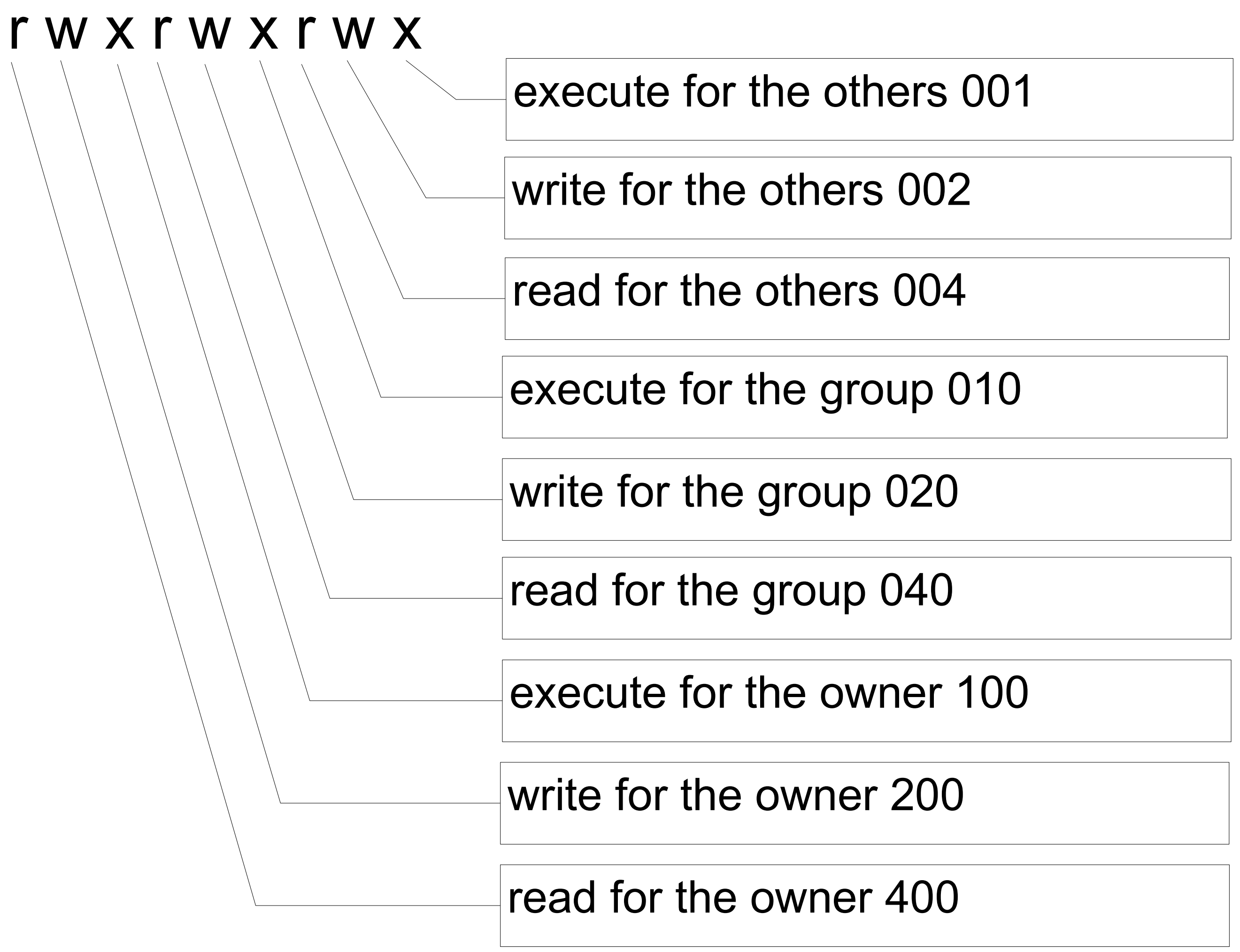
```
jerry@jerryntb:/$ ls -l
total 80
drwxr-xr-x    2 root     bin          4096 May  4  2002 bin
drwxr-xr-x    2 root     root         4096 Dec 29 08:28 boot
drwxr-xr-x    2 root     root         4096 May  4  2002 cdrom
drwxr-xr-x    1 root     root            0 Jan  1  1970 dev
drwxr-xr-x   29 root     root         4096 Feb 16 16:10 etc
drwxr-xr-x    2 root     root         4096 May  4  2002 floppy
drwxr-xr-x    5 root     root         4096 Oct 19 13:05 home
drwxr-xr-x    2 root     root         4096 Dec 22 18:40 jet
drwxr-xr-x    3 root     root         4096 May  4  2002 lib
drwxr-xr-x    2 root     root        16384 Apr  6  2002
lost+found
drwxr-xr-x    3 root     root         4096 May  6  2002 mnt
drwxr-xr-x    5 root     root         4096 May  4  2002 old
drwxr-xr-x    3 root     root         4096 Mar 30  2002 opt
dr-xr-xr-x   73 root     root            0 Feb 16 15:05 proc
drwx--x---    3 root     root         4096 May  6  2002 root
drwxr-xr-x    2 root     bin          4096 May  4  2002 sbin
drwxrwxrwt    7 root     root         4096 Feb 16 14:09 tmp
drwxr-xr-x   17 root     root         4096 Apr  1  2002 usr
drwxr-xr-x   14 root     root         4096 Jan 24  2002 var
```

rights
owner
group
type

Access rights to the filesystem

- if the UID of the object's owner is equal to the effective UID of the process, the owner's permissions are used
- otherwise, if the GID of the object's group is equal to the effective GID of the process or to the GID of any supplementary group of the process, the group's permissions are used
- otherwise the others' permissions are used
- if the effective UID of the process is 0, no access restrictions apply

Access rights to the filesystem



the numbers are octal (base 8 numbering system)

Access rights to the filesystem

- an ordinary file
  - 640 – the owner can read and write, the group can read, the others can do nothing
  - 511 – all can execute, the owner can also read
- a directory
  - 750 – the group can read and use, the owner can also modify, and the others can do nothing
  - 711 – the group and the others can use but not read (list files) or modify the directory

Access rights to the filesystem

- changing the access rights
  - only the owner (and root) can change access rights
  - `chmod rights object ...`
    - rights as an octal number
  - `chmod WhoOpRight[,...] object ...`
    - Who: **u** – owner, **g** – group, **o** – others, **a** = ugo
    - Op: **+** - add, **-** - remove, **=** - set
    - Right: **r**, **w**, **x**
  - `chmod -R ...` - recursively apply to subtree

Access rights to the filesystem

chmod 640 file	chmod u=rw,g=r,o= file
chmod 711 file	chmod u=rwx,go=x file
chmod 711 file	chmod a=rwx,go-rw file
chmod 711 file	chmod a=x,u+rw file

Access rights to the filesystem

- changing the owner
  - only by root
  - `chown [-R] owner[:group] object ...`
    - owner – the username or UID of a user
    - group – the name or GID of a group
- changing the group
  - `chgrp [-R] group object ...`
    - group – the name or GID of a group
  - root can change the group to any group, the owner can change the group to a group he/she is a member of

Access rights to the filesystem

- when a file or a directory is created:
  - owner = effective UID of the creating process
  - group = effective GID of the creating process (SysV) or the group of the parent directory (BSD)
  - rights are defined by the creating process (usually 666 for files and 777 for directories) and modified by the value of **umask** – the rights set in umask are removed from the result
    - **umask** can be set by the `umask` command
    - `umask 022` – the group and others are denied write
    - `umask 077` – the group and others are denied all

Access rights to the filesystem

- Some programs need different rights than that of the user who starts them – e.g. `passwd` needs to write to `/etc/passwd`, or read and write to `/etc/shadow`.
- set-UID permission bit
  - `chmod`: 4000, u+s, `ls -l`: rw**s**r-xr-x
  - the process will have its saved UID and its effective UID equal to the UID of the owner of the executed file
  - the process can switch its effective UID between its real and saved UID

Access rights to the filesystem

- set-GID permission bit
  - `chmod`: 2000, g+s, `ls -l`: rwxrwx**s**r-x
  - the process will have its saved GID and its effective GID equal to the GID of the executed file's group
  - the process can switch its effective GID between its real and saved GID
- set-GID permission bit on directories (SysV)
  - created files and subdirectories will have their group equal to this directory's group and subdirectories will have their set-GID bit set

Access rights to the filesystem

- sticky bit on directories
  - `chmod`: 1000, +t, `ls -l`: rwxrwxrw**t**
  - an object in this directory can be removed only by the owner of the object, by the owner of the directory or by root; the standard access rights are applied as well, i.e. the user must have **w** right to the directory as well
  - /tmp has rights 1777

UNIX directory tree

- /dev
  - character and block devices
- /etc
  - configuration files
- /bin
  - basic system programs for normal users
- /sbin
  - basic system programs for administrators

UNIX directory tree

- /lib
  - basic system shared libraries
- /tmp
  - temporary files
- /boot
  - kernel and other files for boot loader
- /proc
  - system information and access to the running system's parameters

UNIX directory tree

- /var
  - varying files (locks (/var/lock), queues (/var/spool), PID files (/var/run), logs (/var/log), mailboxes (/var/mail, /var/spool/mail), applications' files (/var/lib/application), ...)
- /root
  - root's home directory
- /home
  - home directories for normal users

UNIX directory tree

- /mnt
  - the mount-point for temporary filesystems
- /opt
  - the directory for optional subsystems
- /usr/bin
  - most of the application executables
- /usr/sbin
  - supplemental programs for administration

UNIX directory tree

- /usr/include
  - header files (.h) for C/C++ programs
- /usr/lib
  - libraries (both shared (.so) and static (.a))
- /usr/share
  - files shareable across architectures
- /usr/local
  - hierarchy for locally installed software
- /usr/X11 – X Windows (on some systems)

UNIX directory tree

- /media
  - the directory for mounting removable media (mostly on desktop systems)
- /sys
  - Linux sysfs – special filesystem to access information and control various system devices and drivers

UNIX filesystem

- i-node – data structure with information about a filesystem object
  - type
  - size
  - owner, group
  - permissions
  - the time of the last access, modification, i-node change
  - number of links to the object from directories
  - list of blocks of the file

UNIX filesystem

- directory
  - name
  - i-node number
- types of filesystem objects
  - normal file
  - directory (d)
  - character/block device (c/b)
  - symbolic link (l)
  - pipe/fifo (p)
  - socket (s)

UNIX filesystem

- There can be unlimited number of (hard) links to an object.
- A filesystem object is removed when
  - the number of links from directories = 0 and
  - the number of links from the table of open files = 0.
- Several hard links to the same object differ only in name, other attributes are shared.

UNIX filesystem

- Creating a link to an object:
  - when the object is created
  - using the command `ln`:
    `ln existing dest`
    - created the link `dest`
  - `ln exist1 ... directory`
    - creates links with the same names in the specified directory
  - all links are equivalent, it is impossible to determine how they have been created
  - it is impossible to create a link to a directory

Filesystem objects

- Symbolic link (soft link, symlink)
  - special file containing a path (relative or absolute) to another object
  - common operations except **remove** are performed on the object that the symbolic link points to
  - the **remove** operation removes only the symbolic link, not the primary object
  - symbolic link has no permissions
  - **broken link** – symlink a non-existent object

Filesystem objects

- Creating a symlink:
  - using the command ln:
    `ln -s path dest`
    - creates the symlink dest containing the specified path
  - `ln -s path1 ... directory`
    - creates symlinks in the specified directory with the same names as the specified objects
- Relative vs. absolute path
  - relative path is relative to the symlink
  - absolute path starts with /

Filesystem objects

- Symlink vs. hard link
  - symlink can point to a directory
  - symlink can point to a different filesystem
  - symlink does not prevent you from removing the object – it has no influence on the number of links in the i-node
  - symlink can point to a non-existent object
  - symlink can be differentiated from the primary object

Filesystem objects

- Pipe (fifo)
  - a one-way interprocess communication channel
    - one process opens it for writing, another one for reading
  - unnamed pipe – is not a filesystem object
  - creating:
    ```
    mkfifo [-m mode] name ...
    ```
    - `mode` = permissions as an octal number
    - default permissions: 666

Filesystem objects

- Directory
  - creating:
    ```
    mkdir name ...
    ```
  - deleting (removing):
    ```
    rmdir name ...
    ```
  - only an empty directory can be removed
    - empty directory contains only . and .. records

Filesystem objects

- Block and character devices
  - most devices (except for network interfaces) are represented by special “files”
  - from the kernel's point of view they are identified by a pair of numbers
    - **major number** – a group of devices of a type
    - **minor number** – identifies a single device of the type
  - the special files are usually located in **/dev**
  - creating:
    ```
    mknod [-m mode] name b|c major minor
    ```

Block and character devices

- block device
  - the basic unit is a **block**
  - e.g. disk, CD, floppy disk, ramdisk
  - it can contain a filesystem
- character device
  - the basic unit is a **character** / byte
  - e.g. serial port, terminal, console, printer, tape

Selected character devices

- /dev/null
  - read: an empty file, write: throws away all data
- /dev/zero
  - read: an infinite file of zero bytes
- /dev/full
  - read: /dev/zero, write: full disk
- /dev/random
  - a random generator output; when it runs out of entropy, it blocks until new entropy is gathered

Selected character devices

- /dev/urandom
  - a pseudorandom generator output – does not block
- /dev/tty
  - the current controlling terminal of a process
- /dev/tty1, /dev/tty2, ...
  - virtual consoles
- /dev/console, /dev/tty0
  - the current virtual console

Selected character devices

- /dev/ttyS0, /dev/ttyS, /dev/ttyUSB0, ...
  - serial ports
- /dev/lp0
  - the printer of the first parallel port
- /dev/ptmx, /dev/pts/0, ...
  - pseudoterminal devices
- /dev/tty??, /dev/pty??
  - pseudoterminal devices

Selected block devices

- /dev/sda, /dev/sdb, /dev/sdc, /dev/sdd, ...
  - disks (SCSI, SATA, SAS, USB, IDE with newer drivers)
- /dev/sda1, /dev/sda2, /dev/sda3, /dev/sda4
  - primary partitions of /dev/sda
- /dev/sda5, ...
  - logical partitions of /dev/sda
- /dev/fd0
  - floppy disk

Selected block devices

- /dev/hda, /dev/hdb, /dev/hdc, /dev/hdd, ...
  - IDE disks and CD-ROMs with older drivers
- /dev/sr0, /dev/sr1, ...
  - CD/DVD ROM
- /dev/sg0, /dev/sg1, ...
  - SCSI generic device
- /dev/c0t1d0s2
  - SCSI device – controller 0, target 1, LUN 0, slice 2

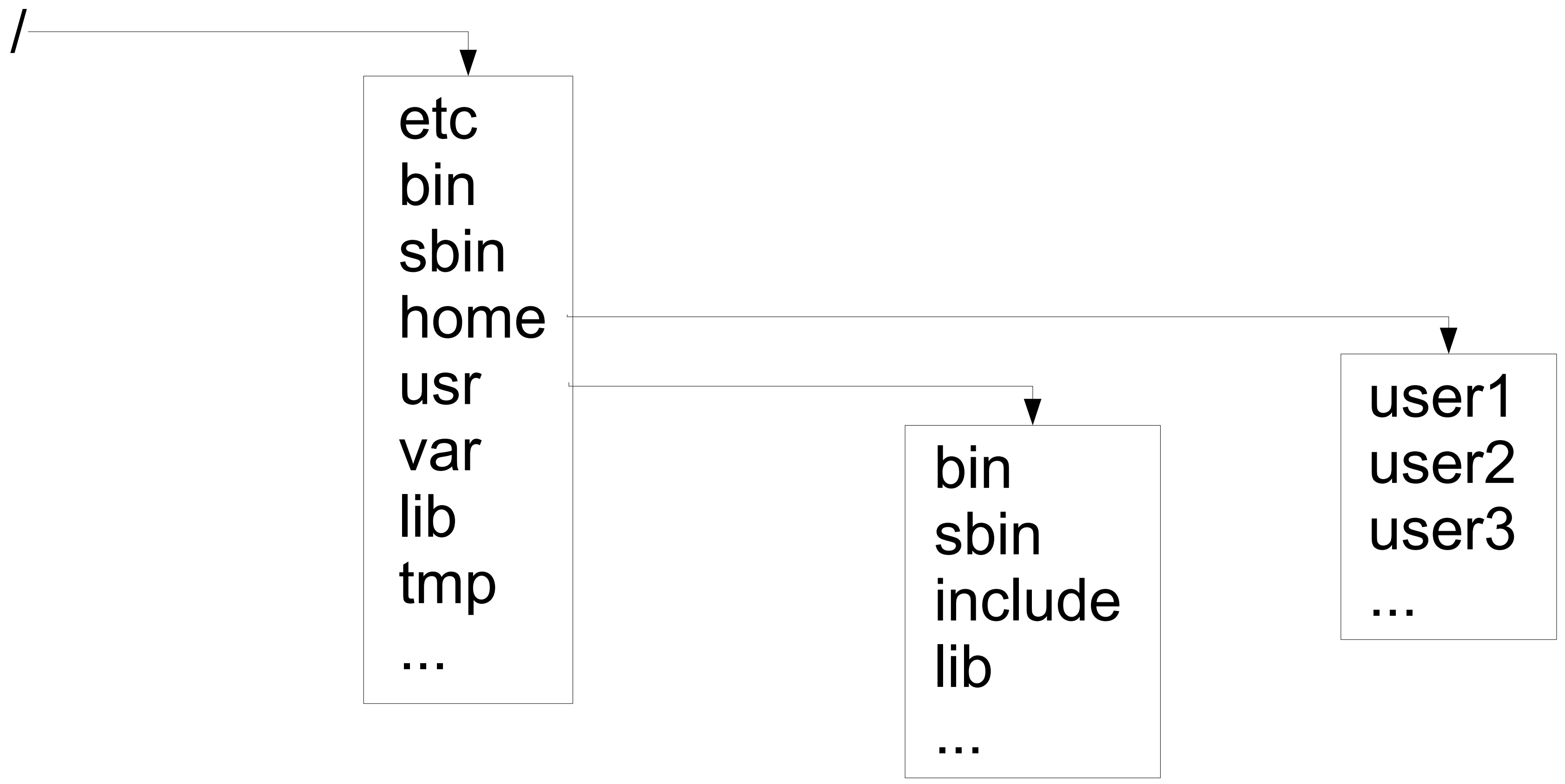
Selected block devices

- /dev/ram0, /dev/ram1, ...
  - ramdisks
- /dev/loop0, /dev/loop1, ...
  - loopback – allows you to access a file as a block device
    `losetup /dev/loop0 file` – associate the file with the loopback block device
    `losetup -d /dev/loop0` – release the loopback device

Mounting filesystems

- single directory tree (or DAG) for file access
- filesystems are mounted on directories (mount points)
- the original content of a mount point becomes temporarily inaccessible – it is replaced with the root directory of the mounted filesystem

Mounting filesystems



Mounting filesystems

- mounting a filesystem (FS)
  `mount [parameters] block_dev dir`
  - mounts the filesystem on the block device to the directory
- `mount -a [-t type]` – mounts all filesystems of the given type specified in `/etc/fstab` without the `noauto` attribute
- `umount {block_dev|dir}` – unmounts the filesystem on the given block device, resp. mounted on the gived directory

Mounting filesystems

- `mount` – shows /etc/mtab – the list of mounted filesystems
  - more exact information can be read from `/proc/mounts`
- options for `mount`:
  - `-t type` – the type of the filesystem
  - `-r,-w` – mount read-only, read-write
  - `-n` – do not update `/etc/mtab`
  - `-o opt1[,opt2,...]` – various options

Mounting filesystems

- The file `/etc/fstab`
  - contains information about filesystems used by `mount`
    - `mount -a, mount directory`
  - lines starting with # are comments
  - fields are separated by spaces and tabs
  - block_dev mount-point type options fs_freq pass_no
    - fs_freq – used by some back-up systems
    - pass_no – the order of checking FS on boot (0 = without)

Mounting filesystems

- options for mounting FS (general)
  - `sync, async` – synchronous / asynchronous writes
  - `[no]atime` – update last access time
  - `[no]auto` – mount by mount -a
  - `[no]dev` – FS can contain block/char. devices
  - `[no]exec` – FS can contain executable files
  - `[no]suid` – FS can contain set-UID a set-GID programs

Mounting filesystems

- `[no]user` – FS can be mounted by a user (using `mount directory`)
  - user implies noexec,nosuid,nodev
- `ro` – FS is read-only
- `rw` – FS is read-write
- `defaults` = `rw,suid,dev,exec,auto,nouser,async`
  - useful in `/etc/fstab`, because the fields cannot be empty
- `remount` – change parameters of a mounted FS

Mounting filesystems

- most common FS types in Linux
  - proc
    - uses no block device (use `none`)
    - an interface to system parameters
    - information about running processes
    - usually mounted on `/proc`
  - tmpfs
    - uses no block device (use `none`)
    - keeps data in memory
    - used for small temporary storage (e.g. for `/dev`)

Mounting filesystems

- devpts
  - uses no block device (use `none`)
  - contains the slave device of a pseudoterminal pair, the master is `/dev/ptmx`
  - usually mounted on `/dev/pts`
  - options
    - uid=value – the owner of the created devices
    - gid=value – the group of the created devices
    - mode=value – permissions for the created devices
- ext2, ext3, ext4
  - common FS for Linux
  - ext3, ext4 are journalling extensions, better resistance

Mounting filesystems

- msdos, vfat
  - FAT type FS (DOS, Windows)
  - msdos – names of 8+3 characters (DOS)
  - vfat – “long names” (Windows)
  - important options:
    - uid=value – the owner of all objects
    - gid=value – the group of all objects
    - umask=value – permissions to be switched off
    - codepage=value – code page for short names – e.g. 437 (US) or 852 (PC Latin2)
    - iocharset=value – the encoding for long name conversion (stored in Unicode) – e.g. iso8859-2 or utf-8

Mounting filesystems

- iso9660
  - standard FS on CD-ROM
  - supports Rock Ridge and MS Joliet extensions
  - can only be read-only
  - important options
    - norock – disables Rock Ridge extensions (they allow objects with UNIX properties – long names, symlinks, permissions, ...)
    - nojoliet – disables Microsoft Joliet extensions
    - uid=value – the owner
    - gid=value – the group
    - mode=value – the permissions (for non Rock Ridge CD)
    - iocharset=value

Example of /etc/fstab

```
/dev/hda5         swap              swap   defaults          0   0
/dev/hda3         /                 ext2   defaults          1   1
/dev/hda6         /mnt/fat          vfat   uid=jerry,umask=077 1   0
/dev/cdroms/cdrom0        /cdrom    iso9660 noauto,user,ro   0   0
/dev/fd0          /floppy           vfat   noauto,user,iocharset=iso8859-2 0
0
/dev/sda1         /jet              vfat   noauto,user,iocharset=iso8859-2 0
0
none              /dev/pts          devpts gid=5,mode=620    0   0
none              /proc             proc   defaults          0   0
none              /proc/bus/usb     usbdevfs defaults        0   0
```

Examples for mount

- `mount -t vfat /dev/sda1 /mnt`
  - mounts the vfat FS from /dev/sda1 to /mnt
- `mount /cdrom`
  - mounts the FS according to /etc/fstab on /cdrom
- `mount -o ro,remount /`
  - makes the root FS read-only
- `mount -o loop,ro -t iso9660 /tmp/image.iso /mnt`
  - connects the file to /dev/loop? and then mounts the FS

Creating filesystems

- `mkfs -t type [fs-options] blok_dev`
  - creates a filesystem of the given type using the program
    `/sbin/mkfs.type`
  - mkfs.ext2 → mke2fs
    - mke2fs /dev/sda1 – creates ext2 FS
    - mke2fs -j /dev/sda1 – creates ext3 FS
  - mkfs.msdos → mkdosfs
    - mkdosfs /dev/sda2 – creates FAT (msdos, vfat) FS
    - mkdosfs -F 32 /dev/sda2 – creates FAT32 FS

System start-up (booting)

- BIOS/firmware loads and starts a boot-loader
- boot-loader loads the kernel of an OS, passes parameters to it and starts it
- kernel:
  - initializes necessary devices
  - mounts a filesystem on /
  - executes `/sbin/init` as the process 1
- init reads `/etc/inittab`

System start-up (booting)

- runlevels (s, 0 – 6)
  - each level is defined by processes that are to run in it
  - common meaning of the runlevels:
    - 0 – shutdown the system
    - 1, s – single user mode
    - 6 – reboot
    - 2, 3, 4 – multi user mode
    - 5 – in Solaris on Suns = turn the power off

System start-up (booting)

- /etc/inittab file
  - lines starting with # are comments
  - fields are separated by colons
  - id:runlevels:action:process
  - id – 1 – 2 characters, unique line identifier
  - runlevels – list of relevant runlevels
  - action – what is to be done
  - process – the command to execute

System start-up (booting)

- actions:
  - **initdefault** – specifies the default runlevel; process is ignored
  - **sysinit** – the process will be started on start-up (before boot and bootwait), init will wait for the termination of the process, runlevels are ignored
  - **bootwait** – the process will be started on start-up, init will wait for its termination, runlevels are ignored (not on all systems)
  - **boot** – like bootwait but init will not wait for the termination of the process

System start-up (booting)

- **wait** – the process will be started on entry to the runlevel and init will wait for its termination
- **once** – the process will be started on entry to the runlevel
- **respawn** – the process will be started on entry to the runlevel and started again after it terminated
- **off** – no action
- **ctrlaltdel** – the process will be started on pressing Ctrl+Alt+Del

System start-up (booting)

- **powerwait** – the process will be started on power failure, init will wait for its termination
- **powerfail** – like powerwait without waiting
- **powerokwait** – the process will be started when power is restored
- **powerfailnow** – the process will be started on low battery signal from UPS
- **ondemand** – the process will be started on demand to enter a special runlevel A, B, C, the real runlevel will not change

System start-up (booting)

- Runlevel can be changed:
  - using `telinit new_runlevel`
  - using `shutdown`
    - -h = 0
    - -r = 6
    - otherwise 1
- On a runlevel change init will kill processes it has started which are not to run in the new runlevel

Start-up scripts structure

- Slackware Linux
  - directory /etc/rc.d:
    - rc.S started as sysinit according to inittab
    - rc.K started on entry to single user mode
    - rc.M started on entry to multi user mode
    - rc.inet1, rc.inet2 initialize networking and some network services
    - rc.local is intended to start locally installed subsystems
    - rc.sshd, rc.httpd, rc.nfsd, ... start relevant subsystems

Start-up scripts structure

- Solaris, Debian Linux, and many others:
  - directory /etc/init.d contains scripts for individual subsystems
  - directories /etc/rcX.d for each runlevel X contain symlinks named like SnnName and KnnName, that are started on entry to the runlevel:
    - nn = priority (order)
    - Name = name of the subsystem
    - S – the script is started with the parameter **start**
    - K – the script is started with the parameter **stop**

Dependency based booting

- ordering of the start-up scripts is determined automatically according to their dependencies
  - comments in the scripts' headers
- `insserv script`
  - adds the script to the set
- `insserv -r script`
  - removes the script from the set

Dependency based booting

```
### BEGIN INIT INFO
# Provides:           boot_facility_1 [ boot_facility_2 ...]
# Required-Start:     boot_facility_1 [ boot_facility_2 ...]
# Required-Stop:      boot_facility_1 [ boot_facility_2 ...]
# Should-Start:       boot_facility_1 [ boot_facility_2 ...]
# Should-Stop:        boot_facility_1 [ boot_facility_2 ...]
# X-Start-Before:     boot_facility_1 [ boot_facility_2 ...]
# X-Stop-After:       boot_facility_1 [ boot_facility_2 ...]
# Default-Start:      run_level_1 [ run_level_2 ...]
# Default-Stop:       run_level_1 [ run_level_2 ...]
# X-Interactive:      true
# Short-Description: single_line_description
# Description:        multiline_description
### END INIT INFO
```

System shutdown

- shutdown command:

```
shutdown [-t sec] [-arkhc] when [message]
```

  - `sec` – time (in seconds) between a request for a process to terminate and its forceful termination
  - `-a` – check whether an authorized user is logged in on the console (listed in `/etc/shutdown.allow`)
  - `-r` – reboot, `-h` – halt, `-k` – send message only
  - `-c` – cancel – cancel the shutdown in progess
  - `message` – displayed on all terminals

System shutdown

- time specification for shutdown:
  - `now` – immediately
  - `hh:mm` – at the specified time
  - `+m` – after m minutes
- if neither -h nor -r, change to the runlevel 1
- alternative syntax on some systems (e.g. Solaris):
  - `shutdown [-g sec] [-i runlevel] [message]`

Job scheduling

- irregular job schedulling – **at**
- `at time`
  - reads commands to be executed from standard input (use Ctrl+D to signal the end of input from a terminal)
  - executes the commands at the specified time using /bin/sh and send the output be e-mail to the user

Job scheduling

- time specification for at:
  - now [+ increment]
  - time [+ increment]
  - time date
  - time = hh:mm | noon | midnight | teatime (16:00)
  - date = name_of_month day [, year] |
    day_of_week | today | tomorrow |
    MM/DD/YYYY | YYYY-MM-DD |
    DD.MM[.YYYY] | DD mmm [YYYY]
  - increment = number minutes|hours|days|weeks|months
    next day|week|month|day_of_week

Job scheduling

- examples:
  - at now + 2 hours – after 2 hours
  - at noon tomorrow – tomorrow at 12:00
  - at 16:00 next monday – next Monday at 16:00
  - at 23:00 2014-04-15 – 15.4.2014 at 23:00
  - at 23:00 04/15/2014 – 15.4.2014 at 23:00
  - at 23:00 15.4.2014 – 15.4.2014 at 23:00

Job scheduling

- listing schedules jobs
  - `atq`
- cancelling a scheduled job
  - `atrm job_id`
- scheduling a job at a time when the system load is low
  - `batch`
    - reads the commands to executed from standard input and executes them when system load drops below 0.8

Job scheduling

- root can use **at** in any case
- using **at** by ordinary users is controlled using two files:
  - if `/etc/at.allow`, exists, `at` can be used only by users listed in it
  - otherwise, is `/etc/at.deny`, exists, `at` can be used by all users **not** listed in it
  - otherwise only root can use **at**
- `atrm` can be used by the owner of the job or by root

Job scheduling

- regular job scheduling – **cron**
- regular jobs execution is controlled by a table with fields separated by spaces
- min hour day month day_of_week command
- a fields can contain a number, a range (2-6), multiple numbers or ranges separated by commas (2-6,8), * (any value)
- the command is executed every minute when the current time matches the specified values

Job scheduling

- examples of specification for cron:
  - `10 6 * * *` = every day at 6:10
  - `10 6 4 * *` = each 4th of a month at 6:10
  - `10 6 * 4 0` = each Sunday of April at 6:10
    - 0=Sunday, 1=Monday, ..., 6=Saturday
  - `* 6 * * *` = every minute between 6:00 – 6:59
  - `10,20 4 * * *` = 4:10 a 4:20
  - `* 7-9 * * *` = every minute between 7:00 – 9:59
  - `*/2 5 * * *` = every other minute between 5:00-5:59 (i.e. 5:00, 5:02, 5:04, ..., 5:58)

Job scheduling

- submitting a table to cron
  - `crontab file [-u username]`
- editing the current table
  - `crontab -e [username]`
- listing the current table
  - `crontab -l [username]`
- deleting the current table
  - `crontab -d [username]`

Job scheduling

- execution of jobs of the cron subsystem is done by the process `crond`
- the tables are stored in `/var/spool/cron/crontabs`
- execution of jobs of the at and batch subsystems is done
  - either by the process – `atd`
  - or by `atrun` which is executed regularly by cron

Networking

- the most widely used means of network communication in UNIX systems (as well as in general) is the TCP/IP protocol family
- a system is connected to networks using *network interfaces*
  - permanent interfaces – e.g. network cards
    - Linux: eth0, Solaris: le0, hme0
  - dial-up connections – e.g. a logical network interface for PPP connection over a modem (ppp0)

TCP/IP basics

- network layer – protocol IP
  - connection less, unreliable
  - provides for the transfer of packets between any two computers (nodes) in a network
  - addresses – 4 B (32 bit) numbers (1.2.3.4)
  - an IP address consists of a *network address* and a *host ID* of the computer within the network
  - *network mask* (*netmask*) specifies the network address part
    - 4B, starts with binary 1s (network), ends with binary 0s (node)
    - 255.255.254.0: the first 23 bits contain the network address

TCP/IP basics

  - netmask can also be specified just by the number of bits comprising the network address (the number of 1s)
- special IP addresses:
  - 127.0.0.0/8 – loopback (interface lo, usually 127.0.0.1) – communication within the host itself
  - host ID = 0 – network address
  - host ID = 1...1 – broadcast – for all hosts in the network
  - 255.255.255.255 – broadcast within local network
  - 224.x.x.x – 239.x.x.x – multicast
  - 240.x.x.x – 255.x.x.x – reserved
  - 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 – private

TCP/IP basics

- examples
  - 158.195.18.0/255.255.255.0 (24)
    - addresses: 158.195.18.1 – 158.195.18.254
    - broadcast: 158.195.18.255, network address: 158.195.18.0
  - 158.195.22.0/255.255.255.128 (25)
    - addresses: 158.195.22.1 – 158.195.22.126
    - broadcast: 158.195.22.127, network address: 158.195.22.0
  - 158.195.22.128/255.255.255.128 (25)
    - addresses: 158.195.22.129 – 158.195.22.254
    - broadcast: 158.195.22.255, network addr.: 158.195.22.128
  - 158.195.16.0/255.255.254.0 (23)
    - addresses: 158.195.16.1 – 158.195.17.254
    - broadcast: 158.195.17.255, network address: 158.195.16.0

TCP/IP basics

- interface dependent protocols are used on link and physical layers
  - in broadcast-type networks (e.g. Ethernet) ARP (address resolution protocol) subsystem is often used to map IP address to link-layer addresses
  - direct communication is possible only between devices connected to the same link-layer network
  - other devices communicate via **routers**

TCP/IP basics

- on transport layer, two protocols are used:
  - TCP (connection oriented, reliable)
  - UDP (connection less, unreliable)
  - both add **port** numbers to IP addresses of the source and destination nodes
  - the quadruplet (source IP address, source port, destination IP address, destination port) uniquely identifies a communication

TCP/IP basics

- routing table
  - determines where to send a packet in the next step based on the destination IP address
  - network address, mask, router addr., interface
  - ordered by network mask (from the most specific to the most general)
  - the first matching line is used
    - if no router is specified, the packet is sent directly using the specified interface to a local network
    - if a router is specified, the packet is sent via the router

TCP/IP basics

– a routing table example

158.195.18.0	255.255.255.0		eth0
127.0.0.0	255.0.0.0		lo
0.0.0.0	0.0.0.0	158.195.18.209	eth0

– a routing table example with 2 interfaces

158.195.18.0	255.255.255.0		eth0
158.195.16.0	255.255.254.0		eth1
127.0.0.0	255.0.0.0		lo
0.0.0.0	0.0.0.0	158.195.16.208	eth1

TCP/IP configuration

- network interface configuration
  - `ifconfig [`*`interface`*`]` – shows the configuration of the specified (or of all active) int.
  - `ifconfig -a` – show the configuration of all int.
  - `ifconfig` *`iface address`* `|` *`options`*`...` configures the interface
  - most common options
    - `netmask` *`mask`* – specifies the netmask
    - `broadcast` *`address`* – specifies the broadcast address

TCP/IP configuration

  - `up` – activates the interface
  - `down` – deactivates the interface
  - `pointopoint` *`address`* – configures the other side's address of a point-to-point link
  - `[-]arp` – turns on/off the ARP subsystem
  - `hw ether` *`address`* – configures link-layer address

- examples
  - `ifconfig eth0 1.2.3.4 netmask 255.255.255.0 broadcast 1.2.3.255` configures an interface
  - `ifconfig eth0 1.2.3.4 netmask 255.255.255.255 pointopoint 5.6.7.8` configures an interface for a point-to-point link

TCP/IP configuration

- routing table management
  - `route [-n]` – lists the routing table (-n will prevent IP address to name conversion)
  - `route add -net` *`dest`* `netmask` *`mask`* `[gw` *`router`*`] [dev` *`iface`*`]` – adds a record for a network to the routing table
  - `route add -host` *`dest`* `[gw` *`router`*`] [dev` *`iface`*`]` adds a record for a single host
  - `route add default [gw` *`router`*`] [dev` *`iface`*`]` adds a default (0.0.0.0/0) route

TCP/IP configuration

  - `route del [-host|-net]` *`dest`* `[netmask` *`mask`*`] [gw` *`router`*`] [dev` *`iface`*`]` removes the record from the table
- enabling the routing in Linux
  - write 1 to `/proc/sys/net/ipv4/ip_forward`
- adding multiple IP addresses to an interface
  - `ifconfig eth0:1 ...` – creates the interface eth0:1 and configures it

Network configuration in Debian

- /etc/network/interfaces
  - auto
    - space separated list of interfaces to configure on start-up
  - allow-hotplug
    - space separated list of interfaces to configure on their creation
  - iface *interface protocol-family method*
    parameters
    ...
    - configuration of the specified interface

Network configuration in Debian

- methods for the family *inet* (i.e. IPv4)
  - loopback
    - for the lo interface
  - dhcp
    - acquires the configuration using DHCP
  - static
    - address
    - netmask
    - broadcast
    - gateway

Network configuration in Debian

- common parameters
  - pre-up *command*
  - up | post-up *command*
  - down | pre-down *command*
  - post-down *command*
- scripts in /etc/network/if-*param*.d/
  - executed after the explicitly specified commands

Network configuration in Debian

- `ifup` *`interface`*
  - configures the specified interface
  - `-a` – all „auto“ interfaces
- `ifdown` *`interface`*
  - deactivates the specified interface
  - `-a` – all „auto“ interfaces

Example /etc/network/interfaces

```
auto lo
iface lo inet loopback

allow-hotplug eth0 eth1

iface eth0 inet dhcp

iface eth1 inet static
            address 192.168.1.2/24
            gateway 192.168.1.1
```

Network services

- `/etc/services` contains the mapping between *service names* and port numbers (and protocols)
  - name port number/protocol alias
- protocol names are mapped to protocol numbers using `/etc/protocols`
- programs providing network services (daemons, servers) either
  - *listen* on a port, waiting for requests, or
  - are started by the “superserver” **inetd**

Network services

- **inetd** is configured in `/etc/inetd.conf`
  - space separated fields
  - name type protocol wait/nowait user prog args
    - name – service name according to /etc/services
    - type – **dgram** or **stream**
    - protocol – **udp** or **tcp**
    - wait = inetd waits for the process's termination before processing new requests on the port
    - nowait = inetd starts a new instance of the program for every request
    - user – username of the user to run the program as

Network services

    - prog – path to the executable file of the program
    - args – arguments (including the zero'th one – the name of the program)

- to perform access control to network services, **tcpd** is often used as a wrapper by **inetd**
  - it uses two files: `/etc/hosts.allow` and `/etc/hosts.deny`
  - hosts.allow is checked first, if no match is found, hosts.deny is checked, and if still no match found, the access is allowed

Network services

- structure of `/etc/hosts.{allow,deny}`
  - list of services : list of clients
  - service:
    - name[@host]
    - ALL
  - client:
    - ALL
    - [username@]host
  - host:
    - IP address, hostname
    - IP address/mask
    - 158.195.
    - .fmph.uniba.sk

Network services

- `netstat` command
  - lists network connections, open ports, routing table, statistics, ...
  - most common switches
    - `-n` – IP addresses and port numbers show numerically (otherwise converted to names)
    - `-a` – list established connections and open ports (listening stream, datagram); otherwise only established connections
    - `-r` – list routing table
    - `-p` – show process ID of the responsible process

IP addresses vs. names

- the file `/etc/hosts`
  - original solution for mapping between IP addresses and hostnames
  - space separated fields, # starts a comment
  - IP_address name aliases
  - complicated assignment of unique names, distribution and updates in large networks
  - usable in small networks

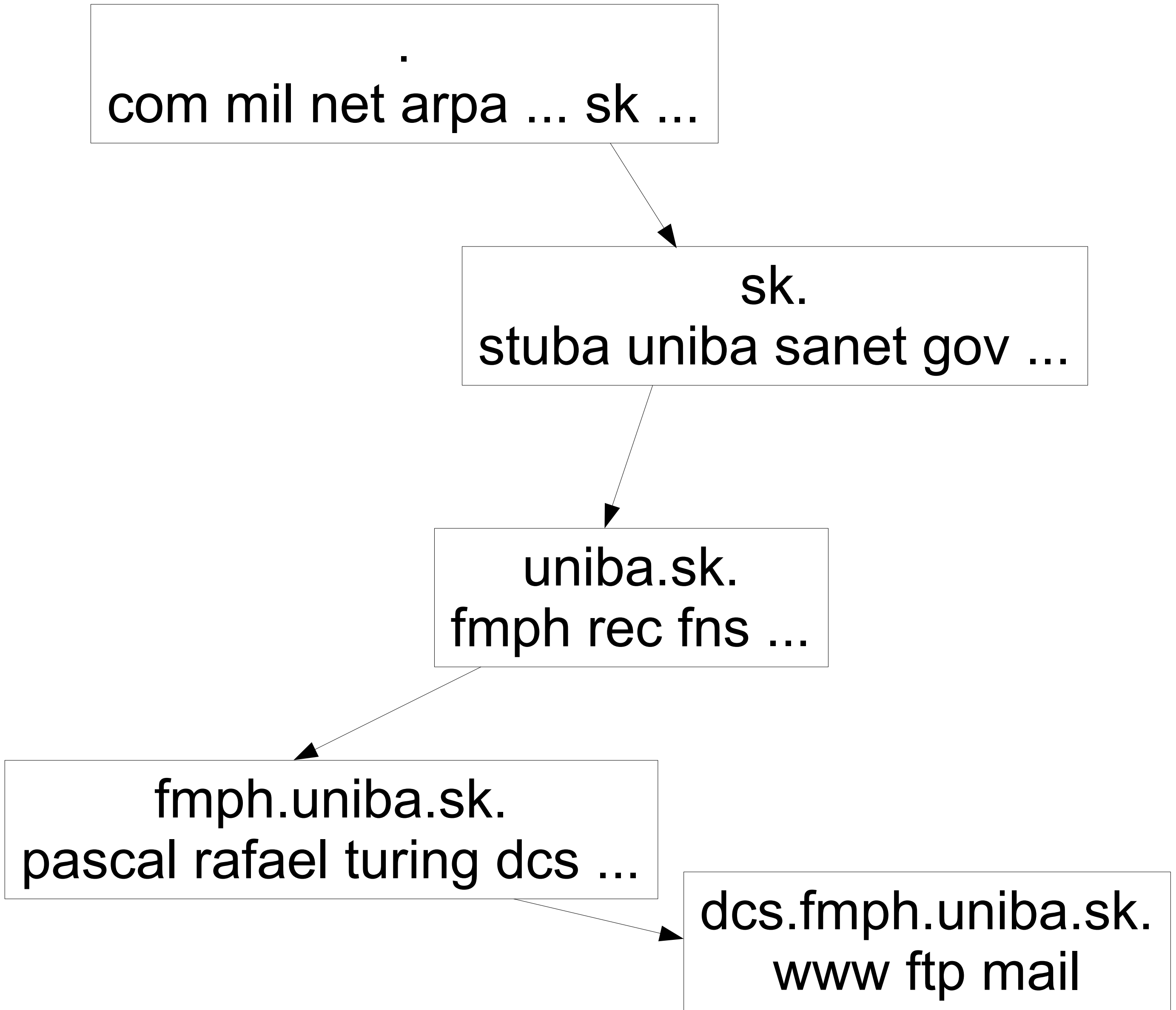
An example of /etc/hosts

```
127.0.0.1        localhost
158.195.16.200   cyril.fmph.uniba.sk        cyril
158.195.87.234   jj-ntb.dcs.fmph.uniba.sk   jj-ntb
158.195.18.163   public.dcs.fmph.uniba.sk   public mail
```

Domain Name System (DNS)

- DNS – the largest distributed database for conversion between domain names and IP addresses
- domain name:
  - name.domain_n.domain_n-1. ... .domain_1
  - does **not** reveal the physical location of the host
  - top-level (1) domains
    - generic: com, org, net, edu, gov, mil, int, biz, info, pro
    - by countries: sk, cz, at, pl, hu, de, uk, ...
- information is provided by DNS servers

Domain Name System (DNS)


.
com mil net arpa ... sk ...
sk.
stuba uniba sanet gov ...
uniba.sk.
fmph rec fns ...
fmph.uniba.sk.
pascal rafael turing dcs ...
dcs.fmph.uniba.sk.
www ftp mail

Domain Name System (DNS)

- domain – each node of the domain tree
- zone – contiguous part of the domain tree for which a name server contains authoritative information
  - entire domain subtree
  - part of a domain subtree – without some of its subtrees that are delegated to another zone (using an NS record)
- DNS server
  - primary – contains zone data in a file
  - secondary – retrieves zone data from a primary server

Domain Name System (DNS)

- zone file format:
  - `$ORIGIN domain` – specifies the suffix for relative domain names
    - absolute domain name in zone file ends with a dot
    - relative domain name – does not end with a dot
  - `$TTL time` – specifies the default time-to-live (in seconds) for records

Domain Name System (DNS)

- domain [TTL] class type data
  - class = `IN` (Internet)
  - domain – domain name (rel. or abs.), for the record
    - @ - the zone's origin
    - space – same as in the previous record
  - type – the type of the record
  - data – the value of the record
    - its syntax depends on the type
  - TTL – the time to keep the record in chaches

Domain Name System (DNS)

- DNS record types
  - `SOA name_serv user.host (serial refresh retry expire neg_ttl)`
    - basic zone information, 1st record
    - name_serv – domain name of the primary DNS server
    - user.host – zone admin's e-mail address (. instead of @)
    - serial – serial number – must be increased on change
    - refresh – the time between regular data reload's from primary to secondary DNS servers

Domain Name System (DNS)

    - retry – delay after unsuccessful refresh
    - expire – if no refresh is successful for this time, zone data are considered invalid (and are not served any more)
    - neg_ttl – TTL for negative responses (non-existing domain)
- `A IP_address`
    - IP address for the domain name
- `CNAME dom_name`
    - the canonical domain name for the domain name of the record (alias)

Domain Name System (DNS)

- `NS name.of.name.server`
  - delegation to another DNS server
- `MX priority mail.server.name`
  - specifies a mail-exchanger for the domain – a computer that receives e-mails for the domain
  - sorted by the priority (lower values = higher priority)
- `PTR domain.name`
  - used in inverse mapping (from IP addresses to domain names)
  - absolute domain name

Domain Name System (DNS)

- IP address to domain name mapping
  - uses a special domain `in-addr.arpa.`
  - to convert IP address A.B.C.D look up a PTR record for `D.C.B.A.in-addr.arpa.`
  - the mappings from a name to an IP address and backwards are independent – there is no mechanism to keep them consistent
    - do not rely on IP address to name conversion for access control

Examples of zone files

```
$ORIGIN dcs.fmph.uniba.sk.
$TTL 36000
@       IN  SOA ns.dcs.fmph.uniba.sk. hostmaster.dcs.fmph.uniba.sk (
                2003042901
                7200
                1800
                86400
                600 )
        IN  NS  ns.dcs.fmph.uniba.sk.
        IN  MX  50 mail
        IN  MX  100 cyril.fmph.uniba.sk.
ns      IN  A   158.195.18.163
mail    IN  A 158.195.18.163
public  IN  A 158.195.18.113
www     IN  CNAME public
-------------------------------------------------------------------------
$ORIGIN fmph.uniba.sk.
dcs     IN  NS  ns.dcs.fmph.uniba.sk.
ns.dcs  IN  A   158.195.18.163
----------------------------------------
$ORIGIN 18.195.158.in-addr.arpa.
@       IN  SOA ...
163     IN  PTR public.dcs.fmph.uniba.sk.
```

```
$ORIGIN 195.158.in-addr.arpa.
18  IN  NS
ns.dcs.fmph.uniba.sk.
```

DNS configuration

- file `/etc/resolv.conf`
  - contains the DNS resolver (client) configuration
  - `domain local_domain`
  - `search dom1 dom2 ...`
    - domain and search are mutually exclusive
    - they specify the suffixes to add to incomplete domain names during lookup
  - `nameserver IP_address`
    - specifies the DNS server for the resolver to send requests to
    - max. 3 records

DNS configuration

- file `/etc/nsswitch.conf`
  - specifies the sources for various mappings used by standard library functions
  - `hosts: files dns`
    - use `/etc/hosts` first, then DNS
- name server (BIND) configuration
  - file `/etc/named.conf`, or `/etc/bind/named.conf`

Network Filesystem (NFS)

- a standard means of file sharing in UNIX systems, based on RPC protocol
- also transfers information about owners, groups and permissions
  - needs common user and group mapping on servers and clients
  - can only be used in a “friendly” environment because of security concerns
- needs the program `portmap`
  - provides port information for services using RPC

Network Filesystem (NFS)

- mounting a share
  - `mount -t nfs server:path dest`
    - mounts the directory `path` on the server `server` into the directory `dest` on the client
- NFS specific mount parameters
  - `fg/bg` – if mounting fails (e.g. due to a network problem), mount will keep trying (fg), or it will keep trying in the background (bg)
  - `noac` – disables attribute caching

Network Filesystem (NFS)

  - `hard` – if the server stops responding, the client's process attempting to access the filesystem will be blocked until the server is OK
  - `soft` – if the server stops responding, the client's process will receive an error indication

- NFS server
  - either a normal process, or a part of kernel
  - `rpc.mountd` – provides the mounting service
  - `rpc.nfsd` – provides the file operations
  - `rpc.statd, rpc.lockd` – provide file locking support

Network Filesystem (NFS)

- Exported (shared) directories are specified in `/etc/exports`
  - each line describes an exported directory and contains the list of allowed clients with some options:
    ```
    /home client1(rw) client2(ro,all_squash)
    ```
  - a client can be specified as
    - a hostname, an IP address, IP address/mask
    - a hostname containing wildcards `*` and `?`

Network Filesystem (NFS)

- options
  - `ro/rw` – read-only/read-write access to the directory
  - `root_squash` – all accesses by the user 0 will be done with the anonymous user's rights
  - `no_root_squash` – root's accesses will be done as root
  - `all_squash` – all accesses will be done as anonymous
  - `async/sync` – (a)synchronous writes

Network Filesystem (NFS)

- Linux kernel contains a built-in NFS server
  - it is controlled using the command `exportfs`
  - `exportfs -r`
    - reads `/etc/exports` and configures the kernel's NFS server
- `rpc.nfsd n`
  - n = the number of threads (default 1, common 8)
  - for kernel NFS server, rpc.nfsd is only a small controlling utility

Other common services

- syslogd
  - central log management
  - /etc/syslog.conf
- sendmail, postfix, exim, qmail
  - mail servers
- apache
  - web server

Printing subsystems

- lpr, lpr-ng
  - printers described in /etc/printcap, configuration in /etc/lpd.conf
  - daemon **lpd**
  - lpr, lp – used to submit print jobs
  - lpq, lpstat – used to show the print queues
  - lprm, cancel – used to cancel a print job
  - /etc/printcap can specify various filters to convert different document formats to a printer's language

Printing subsystems

- CUPS (Common UNIX Printing System)
  - web based administration interface (port 631)
  - supports IPP, lpd, windows driver installation for samba
  - daemon **cupsd**
  - configuration in /etc/cups/
  - many filters, printer drivers, connection methods
  - printer sharing with automatic clients' configuration

Samba

- an implementation of Windows networking protocols for file and printer sharing
- server
  - smbd – provides file and printer sharing services, authentication, ...
  - nmbd – provides Windows name to IP address translation – network browsing
  - swat – web interface for administration

Samba

- client
  - smbclient – ftp like command line client to access Windows shares
  - nmblookup – Windows name lookup tool
  - filesystem cifs
    - mounting Windows directories to the UNIX tree
    - cifs also supports UNIX extensions
    - previously smbfs

Linux Access Control using ACLs

Standard UNIX permissions

- permissions: read, write, execute/use
- subjects: owner, group, others
- assigned to every filesystem object
- problems:
  - too coarse structure of subjects
  - default permissions for new objects cannot be specified

Classic solutions

- crating a suitable set of groups
  - requires root's access
  - complex requirments lead to a high number of groups
- suitable use of umask
  - OK, if we need the same default permissions in all directories
  - problem, if we need different default permissions in various directories

Access Control List (ACL)

- extends the types of subjects that the permissions can be specified for:
  - the owner
  - the assigned group
  - the others
  - a specific user
  - a specific group
- specifies the default permissions (ACL) for new objects in the directory

Access Control List (ACL)

- every filesystem object is assigned a list of entries:
  - the entry's type,
  - an identifier of a user/group,
  - permissions (read, write, execute/use)

Access Control List (ACL)

- entry types:
  - ACL_USER_OBJ – the permissions for the owner
  - ACL_USER – the permissions for a specified user
  - ACL_GROUP_OBJ – the permissions for the object's group
  - ACL_GROUP – the permissions for a specified group
  - ACL_OTHER – the permissions for the others
  - ACL_MASK – the maximal permissions for ACL_USER, ACL_GROUP a ACL_GROUP_OBJ
- 1, 0 or mode, 0 – 1 (1 if a green exists)

Access Control List (ACL)

- Evaluation of the permissions
  - if effective UID of the process = UID of the owner, ACL_USER_OBJ entry is used
  - otherwise, if effective UID of the process = the identifier in an ACL_USER entry, the entry is used after being ANDed with ACL_MASK
  - otherwise all ACL_GROUP_OBJ and ACL_GROUP entries matching the effective GID or a supplementary group of the process are ORed together, ANDed with ACL_MASK and used
  - if no such entry exists, the ACL_OTHER entry is used

Access Control List (ACL)

- Relations to the standard UNIX permissions
  - the owner's permissions correspond to the ACL_USER_OBJ entry
  - the others' permissions correcpond to the ACL_OTHER entry
  - the group's permissions correspond to
    - ACL_GROUP_OBJ, if ACL_MASK is not present
    - ACL_MASK, if ACL_MASK is present
  - changing the standard permissions also changes the corresponding ACL entries, and vice versa

Access Control List (ACL)

- Textual representation of the permissions – the long version
  - one entry per line
  - type:id:permissions
  - type
    - user – ACL_USER or ACL_USER_OBJ (if id=““)
    - group – ACL_GROUP or ACL_GROUP_OBJ (if id=““)
    - mask – ACL_MASK
    - other – ACL_OTHER
  - permissions: 3 characters from {r, w, x, -}

Access Control List (ACL)

- Textual representation – the short version
  - the entries are separated with commas
  - the type can be abbreviated as u, g, m, o
  - dashes (-) in permissions can be left out (but at least one character has to be present)

Displaying the ACL

- `getfacl [-R] object`
  - -R – recursively
  - lists the ACL for the specified object(s) in the long textual form

```
# file: a
# owner: root
# group: root
user::rw-
user:jerry:---
group::r--
group:users:r--
group:testgroup:-w-
mask::rw-
other::---
```

Setting the ACL

- `setfacl [-R] -m acl object`
  - -R – recursively
  - acl – short textual form of the ACL specification
  - adds/modifies the specified ACL entries
- `setfacl [-R] -x acl object`
  - removes the specified entries from the ACL
  - acl – the specifcation of the entries without the permissions
- `setfacl [-R] -b object`
  - removes all entries from the ACL

Setting the ACL

- other options for setfacl
  - -n – do not recalculate the mask
    - setfacl by default, if the mask is not explicitly specified, calculates and sets the mask as OR of all ACL_USER, ACL_GROUP and ACL_GROUP_OBJ entries
- `setfacl --restore=file`
  - sets the ACL according to the file in the format of getfacl output
  - used to restore ACLs saved using getfacl

Default ACL

- A directory can also have a default ACL for new objects
  - a new object's ACL is initialized from the default ACL of the directory
  - the permissions not requested on creation are removed from the entries corresponding to the standard UNIX permissions
- If the directory has no default ACL
  - a new object's ACL will contain the entries corresponding to the standard permissions initialized according to the requested permissions and umask

Setting the default ACL

- `setfacl -d -m acl directory`
  - acl as in the normal ACL
- `setfacl -m defacl directory`
  - the entries of defacl have the prefix **default:** or **d:**
- `setfacl -d -x acl directory`
- `setfacl -x defacl directory`
- `setfacl -k directory`
  - removes the default ACL

Mounting a filesystem with ACL support

- When using mount, or on the corresponding line of /etc/fstab the **acl** option is used:
  - `mount -t ext3 -o acl /dev/sda2 /home`
- ACLs are supported in Linux for ext3 since the 2.6.x kernel

Examples

```
root@lubka:/tmp/x# umask
0027
root@lubka:/tmp/x# mkdir d
root@lubka:/tmp/x# getfacl d
# file: d
# owner: root
# group: root
user::rwx
group::r-x
other::---
```

```
root@lubka:/tmp/x# setfacl -m g:testgroup:rwx,default:g:testgroup:rwx d
root@lubka:/tmp/x# getfacl d
# file: d
# owner: root
# group: root
user::rwx
group::r-x
group:testgroup:rwx
mask::rwx
other::---
default:user::rwx
default:group::r-x
default:group:testgroup:rwx
default:mask::rwx
default:other::---
```

Examples

```
root@lubka:/tmp/x# touch d/f
root@lubka:/tmp/x# mkdir d/d2
root@lubka:/tmp/x# getfacl d/f
# file: d/f
# owner: root
# group: root
user::rw-
group::r-x                    #effective:r--
group:testgroup:rwx      #effective:rw-
mask::rw-
other::---
```

```
root@lubka:/tmp/x# getfacl d/d2
# file: d/d2
# owner: root
# group: root
user::rwx
group::r-x
group:testgroup:rwx
mask::rwx
other::---
default:user::rwx
default:group::r-x
default:group:testgroup:rwx
default:mask::rwx
default:other::---
```

Firewalling in Linux

The task of a firewall

- on an end host
  - access control to network services
  - restriction of outgoing communication
- on a router
  - restriction of communication between networks
  - network address translation (NAT)
    - allowing communication from networks that use private IP addresses

Types of firewalls

- stateless firewall
  - each packet is handled separately
    - based on the network and transport layer headers
  - low demands for CPU time and memory
  - only simple security policies
- stateful firewall
  - keeps information about “connections”
  - higher demand for memory and CPU time
  - more complex policy can be enforced

Firewall in Linux

- a part of the kernel
  - **netfilter** subsystem
- stateful firewall
  - implements connection tracking
    - TCP, UDP
    - helper modules for some problematic application layer protocols (e.g. FTP)
- also a stateless firewall
  - without using connection tracking

Firewall in Linux

- driven by tables (IP tables)
  - filter
    - packet filtering
  - nat
    - network address translation (NAT)
  - mangle
    - manipulation with some packet attributes

Firewall in Linux

- a table contains chains of rules
  - filter
    - INPUT – for incoming packets
    - OUTPUT – for outgoing packets
    - FORWARD – for forwarded packets
  - nat
    - PREROUTING – before routing (forwarded packets)
    - OUTPUT – locally generated packets before routing
    - POSTROUTING – after routing (just before transmission)
  - user-defined chains can be added

Firewall in Linux

- chains contain rules
  - test
  - target
    - ACCEPT – accept the packet for further processing
    - DROP – drop (discard) the packet (silently)
    - a name of a user-defined chain
      - contains using the rules of the specified chain
    - RETURN – continue with rules in the previous chain
    - other
- predefined chains also have a default target

Firewall in Linux

- application of the rules
  - if the test matches, the specified target is used
  - otherwise, the next rule is applied (evaluated)
  - when the end of a user-defined chain is reached, the next rule in the previous chain (i.e. the chain that referenced the user-defined one) is applied
  - at the end of a predefined chain, the default target is used

Firewall in Linux

- command `iptables`
  - `-t table` – the table to show/modify
  - `-A chain rule` – adds a rule at the end of the chain
  - `-I chain [position] rule` – inserts a rule
  - `-D chain position` – removed a rule
  - `-L [chain]` – lists rules (of a chain)
  - `-F [chain]` – removes all rules (from a chain)
  - `-N chain` – defines a new chain

Firewall in Linux

  - `-X [chain]` – removes a chain
  - `-P chain target` – sets the default target for a chain
- rules (! negates the test)
  - `-p [!] protocol` – transport layer protocol
  - `-s [!] address/mask` – source IP address
  - `-d [!] address/mask` – destination IP address
  - `-i [!] interface` – incoming interface
  - `-o [!] interface` – outgoing interface
  - `-j target` – the rule's target

Firewall in Linux

- extension of tests for TCP protocol
  - `--sport [!] port[:port]` – source port, range of ports
  - `--dport [!] port[:port]` – destination port, range of ports
  - `--tcp-flags [!] mask value`
    - SYN, ACK, FIN, RST, URG, PSH, ALL, NONE
  - `[!] --syn`
    - --tcp-flags SYN,RST,ACK,FIN SYN

Firewall in Linux

- extension of tests for UDP protocol
  - `--sport [!] port[:port]` – source port, range of ports
  - `--dport [!] port[:port]` – destination port, range of ports

Firewall in Linux

- extensions of tests for stateful firewall
  - `-m state` – loads the extension module **state**
  - `--state state` – tests the state of a connection
    - NEW – a new packet – the first packet of a connection
    - ESTABLISHED – a packet that belongs to an existing connection
    - RELATED – the first packet of a connection related to an existing connection (i.e. expected due to an existing connection)
    - INVALID – a packet that cannot be identified

Firewall in Linux

- some other targets
  - `REJECT [--reject-with type]` – drops the packet and responds with an ICMP error message
    - icmp-net-unreachable
    - icmp-host-unreachable
    - icmp-port-unreachable
    - icmp-proto-unreachable
    - icmp-net-prohibited
    - icmp-host-prohibited
    - icmp-admin-prohibited

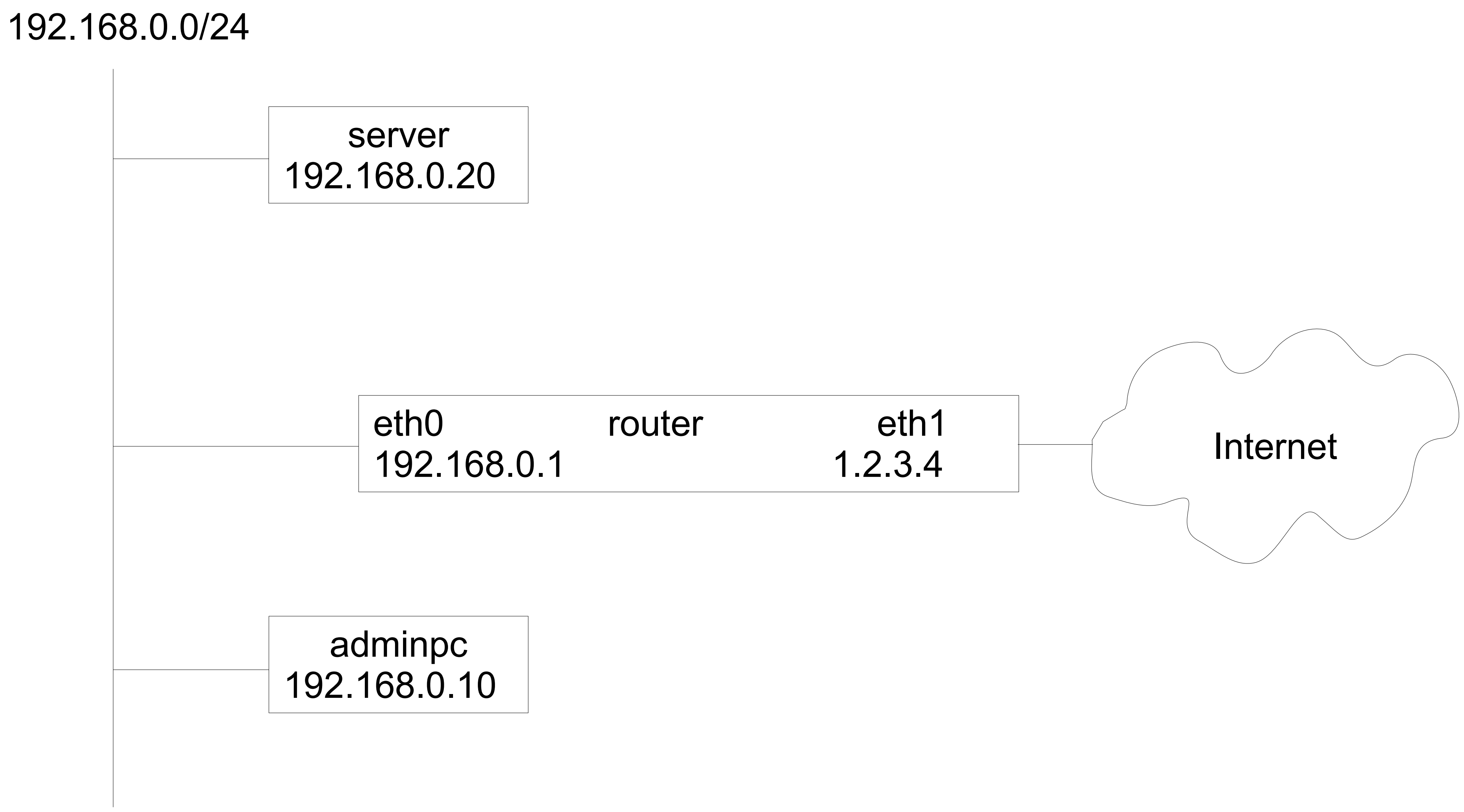
Firewall in Linux

- `REDIRECT [--to-ports port[-port]]`
  - redirects the packet to the local address to the specified port (or a range of ports)
  - used to implement transparent proxy servers
- `SNAT –to-source ipaddr[-ipaddr]`
  - rewrites the source address to the specified one (in POSTROUTING chain in the nat table)
  - following packets are modified in the same/opposite way
- `DNAT –to-destination ipaddr`
  - rewrites the destination address (PREROUTING, OUTPUT in the nat table)
  - following packets are modified in the same/opposite way

An example – end station protection

```
iptables -F
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -s 192.168.0.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

An example – a router



An example – a router

```
iptables -F
iptables -X
iptables -N outgoing
iptables -N server
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -s 192.168.0.10 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -s 192.168.0.0/24 -i ! eth0 -j DROP
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -d 192.168.0.20 -j server
iptables -A FORWARD -s 192.168.0.0/24 -j outgoing
iptables -A outgoing -p tcp --dport 25 -j DROP
iptables -A outgoing -j ACCEPT
iptables -A server -p tcp --dport 80 -j ACCEPT
iptables -A server -p tcp --dport 443 -j ACCEPT
iptables -A server -p udp --dport 53 -j ACCEPT
iptables -A server -p tcp --dport 53 -j ACCEPT
```

An example – a router

```
iptables -t nat -F
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth1 -j SNAT
  --to-source 1.2.3.4
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT
  --to-destination 192.168.0.20
iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT
  --to-destination 192.168.0.20
iptables -t nat -A PREROUTING -p tcp --dport 53 -j DNAT
  --to-destination 192.168.0.20
iptables -t nat -A PREROUTING -p udp --dport 53 -j DNAT
  --to-destination 192.168.0.20
```

FUSE – Filesystem in Userspace

FUSE – Filesystem in Userspace

- Basic idea
  - to allow implementation of a filesystem using an ordinary process
  - a kernel module passes requests from a mounted filesystem to the process
    - module fuse
  - the process implements the filesystem operations

What is FUSE good for?

- easier to implement filesystem “drivers”
  - it is easier to write an ordinary program than to write a kernel module
- it allows normal users to mount filesystem to the tree
  - without specifying fixed parameters in /etc/fstab

FUSE examples

- sshfs
  - allows users to mount a remote computer's filesystem accessible via SFTP protocol
  - mounting

    ```
    sshfs usename@host:path mountpoint
    ```
  - unmounting

    ```
    fusermount -u mountpoint
    ```

How does FUSE work?

- kernel module fuse
  - provides the kernel-side implementation of a filesystem
- libfuse
  - the library used by the process implementing the filesystem
- /dev/fuse
  - character device used for communication between the process and the kernel module
- fusermount
  - helper program to support (un)mounting

How does FUSE work?

- sshfs creates a connection to the remote server
- it passes function pointers of functions implementing individual operations to libfuse
- it runs fusermount using libfuse
- fusermount (set-uid) mounts the filesystem
- requested operations in the mounted filesystem are passed using /dev/fuse to sshfs
- sshfs performs the operation using SFTP and returns the result

FUSE and permissions

- a process that wants to mount a filesystem to a mountpoint must have full access permissions to the mountpoint
- fuse ignores permission checks in the mounted filesystem by default – it leaves the access control up to the process implementing the filesystem
- using the **-o default_permissions** option the standard access control checks can be enabled in the kernel

FUSE and permissions

- the mounted filesystem is inaccessible for other users by default, not even for root
- using the **-o allow_other** option **root** can allow access by other users
  - in /etc/fuse.conf the **user_allow_other** option can be specified to allow normal users to use **allow_other** and **allow_root** (**allow_root** allows access by root)

Some useful FUSE filesystems

- sshfs
  - sftp/ssh
- ntfsmount
  - nfts
- fusedav
  - WebDAV
- encfs
  - encrypted FS
- fuseiso
- fusesmb
  - SMB/CIFS
- gphotofs
  - PTP
- mtpfs
  - MTP
- obexfs
  - OBEX

Linux booting

The booting procedure

- BIOS loads and starts a boot loader
- The boot loader loads
  - the system kernel
  - CPIO archive with the contents of the initial root filesystem (initramfs)
- The boot loader starts the kernel (and passes parameters to it)
- The kernel extracts initramfs to / and starts **/init**
- **/init** mounts the real root filesystem and starts **/sbin/init**

Why initramfs?

- it used to work without it earlier
  - the kernel had to contain all drivers needed to access the root filesystem
- today the kernel is usually universal and the drivers (in the form of kernel modules) are loaded from initramfs before mounting the real root filesystem
- this approach enables using more complicated setup needed to access the root filesystem
  - RAID, LVM, encryption, ...

Creating/updating initramfs

- update-initramfs -u
  - updates the initramfs for the current kernel
- update-initramfs -c -k kernel_version
  - created a new initramfs for the given kernel
- configuration
  - /etc/initramfs-tools/
    - initramfs.conf
    - modules
    - ...

Useful kernel parameters

- init=...
  - file to use instead of /sbin/init
- rdinit=...
  - file to use instead of /init
- ro, rw
  - whether the root filesystem is to be mounted ro/rw
- root=...
  - device containing the root filesystem
- run-level number / single / emergency

Useful kernel parameters

- root=
  - /dev/sda2
  - UUID=20282ab2-2692-4734-8806-f08e52171c0e
  - LABEL=root
  - /dev/nfs
    - nfsroot=[<server-ip>:]<root-dir>
    - ip=<client-ip>:<server-ip>:<gw>:<netmask>:<hostname>:<device>:<autoconf>

How does BIOS load a boot loader?

- BIOS reads 1st disk sector (MBR) and runs it
  - the base part of a boot loader must fit to 446B
  - the rest of the sector contains the partition table (4x16B)
- „standard“ PC boot loader in MBR loads 1st sector of the **active** partition and runs it
  - it loads and runs either
    - the kernel of an OS
    - or the rest of a larger boot loader

Examples of boot loaders capable of loading Linux

- syslinux
  - FAT (e.g. on a USB stick)
- isolinux
  - ISO9660 (CD)
- pxelinux
  - network booting (PXE)
- lilo
  - ext2/3 (+MBR)
- grub
  - various filesystems (+MBR)

syslinux

- configuration in **syslinux.cfg**
- it allows the user to specify parameters for the kernel on a command line
- it can be used to create a simple boot menu

```
default inst

label inst
  kernel inst
  append initrd=instrd.gz
```

isolinux, pxelinux

- isolinux
  - configuration in **isolinux.cfg**
    - same as syslinux
- pxelinux
  - configuration in the directory **pxelinux.cfg**
    - same as syslinux, individual files for client computers
  - loaded using TFTP from the server specified using DHCP
  - allows booting of a disk-less computer
    - with the root filesystem typically mounted via nfs

lilo

- the base part can be in MBR or in an ext2/3 FS
- the rest in ext2/3
- it records the files' positions using sector numbers
  - a problem arises when the files are moved
- configuration in **/etc/lilo.conf**
  - when changes, lilo must be reinstalled
- installation using the command **lilo**

lilo

- can load and run another boot loader (chain loading)
  - it can boot various OS's in this way
- supports creation of boot menus
- allows the user to specify kernel parameters
- can modify the parition type
  - can be used to „hide“ partitions

lilo

```
image = /boot/vmlinuz
  root = /dev/sda2
  label = Linux
  initrd = /boot/initrd.gz
  read-only

other=/dev/sda1
  label=windows
```

grub

- the base part can be in MBR or in 1st sector of a partition
- the 2nd part can be between MBR and the 1st partition or in a partition
  - when in a partition, it is sensitive to changing the position of the files
- configuration and other parts (modules) are in files in the filesystem
  - grub has modules supporting various filesystems (ext2/3/4, fat, ntfs, iso9660, ...)

grub

- can load and run another boot loader (chain loading)
  - can boot other OS's in this way
- supports boot menu creation
- allows the user to specify kernel parameters
- can modify the partition type
  - can be used to „hide“ partitions

grub

- configuration file **/boot/grub/grub.cfg**
  - generated using the command **update-grub** or **grub-mkconfig**
  - configuration parameters
    - /etc/default/grub
  - scripts used during the configuration file generation
    - /etc/grub.d/

grub

```
menuentry 'Debian GNU/Linux, with Linux 3.2.0-0.bpo.3-686-pae'  {
        insmod part_msdos
        insmod ext2
        set root='(hd0,msdos3)'
        search --no-floppy --fs-uuid --set 20282ab2-2692-4734-8806-f08e52171c0e
        echo    'Loading Linux 3.2.0-0.bpo.3-686-pae ...'
        linux   /boot/vmlinuz-3.2.0-0.bpo.3-686-pae root=UUID=20282ab2-2692-4734-8806-
f08e52171c0e ro
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-3.2.0-0.bpo.3-686-pae
}
```

systemd

systemd

- global service manager
  - replacement for /sbin/init
- per user service manager (systemd --user)
  - can run services as the logged in user
- various activation mechanisms
  - sockets, timers, ...

systemd configuration

- /etc/systemd/system.conf (user.conf)
  - various default values (timeouts, limits, ...)
- unit files
  - /etc/systemd/system/, /run/systemd/system, (/usr)/lib/systemd/system/
  - ~/.config/systemd/user/, /etc/systemd/user/, /run/systemd/user/, /usr/lib/systemd/user/

systemd unit file

- text file describing
  - a service (a replacement of init)
  - a socket (a replacement of inetd)
  - a device
  - a mountpoint (a replacement of /etc/fstab)
  - an automount point
  - a swap file/partition
  - a start-up target
  - a watched path
  - a timer
  - ...

systemd unit file

- [Unit]
  Key=*Value*
  ...
  [Install]
  Key=*Value*
  ...
  [*Othersection*]
  Key=*Value*
  ...

systemd unit file [Unit] section

- Description=
  - a textual description of the unit
- Documentation=
  - a URI list of documentation (space separated)
- Wants=
  - a list of units that will also be started when this unit is started (but they may fail)
  - also the directory *unit*.wants may contain symlinks to wanted unit files

systemd unit file [Unit] section

- Requires=
  - stronger version of Wants – it the required units fail to start (and the current one is ordered after the failing one), the current unit will not be started
  - if the required unit is stopped, the current unit will also be stopped
  - also the directory *unit*.requies may contain symlinks to required unit files

systemd unit file [Unit] section

- Requisite=
  - similar to Require, but the listed units must be already started before this one, otherwise this one will fail
- BindsTo=
  - even stronger dependency than Requires
  - if the other unit becomes inactive for any reason, this one will be stopped
- PartOf=
  - when the other unit is stopped/started, this one will be as well

systemd unit file [Unit] section

- Conflicts=
  - if this unit is stared, the other one will be stopped and vice versa
- Before=, After=
  - specify the ordering of units when being started (inverse when being stopped)
  - without ordering, units are startes/stopped simultaneously
- ...

systemd unit file [Install] section

- specifies what to do when a unit is enabled/disabled
- WantedBy=, RequiredBy=
  - a link to this unit will be created in .wants/.requires directory of the other unit
  - usually used to create a dependency for a unit of the target type
- Also=
  - a list of units to be also enabled/disabled

systemd service unit

- uses .service suffix, contains [Service] section
- Type=
  - simple
  - exec
  - forking
  - oneshot
  - dbus
  - notify
  - idle

systemd service type

- simple, exec
  - the service becomes started as soon as a process is created (even before its new binary is exec-ed) or as soon as the new program is exec-ed
- forking
  - it is expected that the service process will fork and exit when it is ready
- oneshot
  - it is considered started when the main process exits

systemd service unit

- RemainAfterExit=yes|no
  - if yes, the service is considered still active after its process terminates (usefull for oneshot)
- PIDFile=
  - specifies the file where the service process writes its PID
- ExecStart=
  - the command to start the service

systemd service unit

- ExecReload=
  - the command to reload the service's configuration
- ExecStop=
  - the command to stop the service
- TimeoutStartSec=, TimeoutStopSec=
  - specifiy the timeout
- Restart=no|on-success|on-failure|on-abnormal|on-abort|always
  - specifies when the service it to be automatically restarted

systemd service unit

- WorkingDirectory=
  - set the working directory of the service
- User=, Group=
  - set the user/group the service runs as
- SupplementaryGroups=
  - set the list of supplementary groups for the process
- Umask=
  - set the umask value

systemd service unit

- RuntimeDirectory=, StateDirectory=, CacheDirectory=, LogsDirectory=, ConfigurationDirectory=
  - systemd will create the directories under standard locations (/run/, /var/lib/, /var/cache/, /var/log, /etc)
- Environment=
  - list of env. variable assignments
- EnvironmentFile=
  - the file containing the env. variable assignments

systemd service unit

- StandardInput=
  - null
  - file:*path*
  - socket
- StandardOutput=, StandardError=
  - inherit
    - duplicate the StandardInput
  - null
  - file:*path*
  - append:*path*
  - socket
  - journal

systemd socket unit

- uses .socket suffix, contains [Socket] section
- needs a service unit of the same name
- ListenStream=, ListenDatagram=
  - specifies the IP address and port number to listen on (e.g. 0.0.0.0:80)
- Accept=yes|no
  - if yes, the connection is accepted and the connected socket is passed to the service

systemd mount unit

- uses .mount suffix, contains [Mount] section
- can be used as a replacement of /etc/fstab
- usually generated from /etc/fstab
- What=
  - specifies the device to mount
- Where=
  - specifies the mountpoint path (must match the name of the unit)
- Type=, Options=

systemd automount unit

- uses .automount suffix, contains [Automount] section
- monitors a path and when it is to be accessed, automatically starts a corresponding mount unit
- Where=
  - the path to monitor/mount
- DirectoryMode=
  - access rights for the directory to be created
- TimeoutIdleSec=
  - unmount after idle time

systemd swap unit

- uses .swap suffix, contains [Swap] section
- similar to mount but for swap devices
- usually generated from /etc/fstab
- What=
- Options=

systemd time unit

- uses .timer suffix, contains [Timer] section
- starts the corresponding service when the timer elapses
- OnActiveSec=, OnBootSec=, OnStartupSec=
  - will start the service specified time after timer activation/boot/systemd start
- OnUnitActiveSec=, OnUnitInactiveSec=
  - will start the service specified time after it was last activated/deactivated

systemd time unit

- OnCalendar=
  - will start the corresponding unit on calendar events (similar to cron)
  - weekday year-month-day hh:mm:ss
  - Mon,Wed *-12-01..20
    - every Monday or Wednesday in December between 1st and 20th every year
- Unit=
  - the unit to start (defaults to the same name as the timer unit)

systemd target unit

- groups services, represents various well known points in the start-up sequence, etc.
- typically uses Wants= or Requires= to start services
- systemd.unit= kernel command line parameter can be used to boot to a specific target instead of default.target

systemd target unit

- multi-user.target
- poweroff.target
- reboot.target
- ctrl-alt-del.target
- shutdown.target
  - in general, services should use Conflicts=shutdown.target to get stopped on shutdown

systemctl

- systemctl list-units
  - lists all loaded units
- systemctl start|stop|restart|reload *service*
  - starts/stops/restarts/reloads the specified service
- systemctl status *unit*
  - show status of the unit

systemctl

- systemctl list-unit-files
  - list unit files
- systemctl enable|disable *unit*
  - enables/disables the unit using its [Install] section
- systemctl mask|unmask *unit*
  - masking prevents the unit from being used (disabled unit can be manually started, masked unit can not be started)

systemctl

- systemctl daemon-reload
  - reload systemd's configuration, regenerate auto-generated units, reload unit files (e.g. after modification), ...
- systemctl halt|poweroff|reboot
- systemctl suspend|hibernate
- systemctl --user ...
  - talk to user's systemd instance instead of the global systemd instance

journalctl

- displays the systemd's journal
  - -u *unit*
    - show only records corresponding to the unit
  - -r
    - show newest records first (reverse order)