Linux 101

… for .NET Devs

Oliver Sturm • @olivers • oliver@oliversturm.com

DevExpress®

Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for over 25 years
- Contact: oliver@oliversturm.com

Agenda

- That Linux Thing
- Getting Started with Linux
    - Shells, Command Lines and Commands
    - File Systems and Permissions
    - Users and Processes
    - Editing and Configuring
    - Packages
- Creating a .NET Core App
- Setting Up a Runtime Environment

On Day 1...

```
From: Linus Benedict Torvalds
Date: August 25 1991
Subject: What would you like to see most in minix?

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. ...

PS. ... It is NOT protable (uses 386 task switching etc), and it
probably never will support anything other than AT-harddisks, as
that's all I have :-(.
```

Full thread: http://osturm.me/torvalds-linux-announcement

On Day 1...

```
From: Linus Benedict Torvalds
Date: August 25 1991
Subject: What would you like to see most in minix?

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. ...

PS. It is NOT portable (uses 386 task switching etc), and it
probably never will support anything other than AT-harddisks, as
that's all I have :-(.
```

Full thread: http://osturm.me/torvalds-linux-announcement

By the Way

Linus doesn't mention it, but his new OS was going to be called *Freax* at this point.

On Day 9658 (Feb 1st 2018)...

- 37-67% of *Web Servers* run Linux (April 2017).
  On the top 1,000,000 domains, it's 96%.
- Since November 2017, *all 500 fastest super-computers* (TOP500 project) run Linux
- IBM runs Linux on *Mainframes* (System z), marketshare ~28% (December 2008)
- 70% of *Mobile Devices* run Linux (Android, December 2017)
- 29% of *Embedded Devices* run Linux (Android and others, March 2012)

On Day 9658 (Feb 1st 2018)...

- 37-67% of *Web Servers* run Linux (April 2017).
  On the top 1,000,000 domains, it's 96%.
- Since November 2017, *all 500 fastest super-computers* (TOP500 project) run Linux
- IBM runs Linux on *Mainframes* (System z), marketshare ~28% (December 2008)
- 70% of *Mobile Devices* run Linux (Android, December 2017)
- 29% of *Embedded Devices* run Linux (Android and others, March 2012)

In all those groups there are large percentages of other Unix-like systems that are not Linux (macOS, iOS, BSD, PlayStation, QNX...)

On Day 9658 (Feb 1st 2018)...

- 37-67% of *Web Servers* run Linux (April 2017).
  On the top 1,000,000 domains, it's 96%.
- Since November 2017, *all 500 fastest super-computers* (TOP500 project) run Linux
- IBM runs Linux on *Mainframes* (System z), marketshare ~28% (December 2008)
- 70% of *Mobile Devices* run Linux (Android, December 2017)
- 29% of *Embedded Devices* run Linux (Android and others, March 2012)

In all those groups there are large percentages of other Unix-like systems that are not Linux (macOS, iOS, BSD, PlayStation, QNX...)

And finally:

- 2% of *Desktop and Laptop machines* run Linux (December 2017)

Amazing Scale and Diversity

- ~30 supported Processor Platforms
- An individual kernel release has > 1000 contributors, ~10000 patches, changing ~3500 lines of code per day.
- Latest kernels have > 20M lines of code. Also several hundred swear words. :)

Amazing Scale and Diversity

- ~30 supported Processor Platforms
- An individual kernel release has > 1000 contributors, ~10000 patches, changing ~3500 lines of code per day.
- Latest kernels have > 20M lines of code. Also several hundred swear words. :)

This is not from the Linux kernel, but funny anyway :)

```
// you can't fix this, but please increment the counter
// below if you try.
// hours wasted here: 56
```

So what have we got?

- Linux is the *kernel*, though the project also includes *drivers*, *filesystems* and other components
- Other parties maintain drivers, *system components* and *application software*
  - Independent system components: *shell tools*, *graphical display servers*, *package management*, ...
- Yet others create *distributions*, with *installers*, *releases*, *updates* and *support lifecycles*
  - Some distributions offer *commercial SLAs*
- Sometimes Linux is *integrated* elsewhere, for instance in embedded systems, or in Windows
- Sometimes Linux is *derived from*, e.g. with Android

As a .NET Dev, why should I give a _?

- It's a *great deployment platform*
  - Fast
  - Performant
  - Scaleable
  - Free of license cost
- It's an *enthusiast/advanced user platform*
- It's small. It's big. It's stable. It boots fast. It's consistent. It's versatile. It's standardized. It's customizable.

As a .NET Dev, why should I give a _?

- It's a *great deployment platform*
  - Fast
  - Performant
  - Scaleable
  - Free of license cost
- It's an *enthusiast/advanced user platform*
- It's small. It's big. It's stable. It boots fast. It's consistent. It's versatile. It's standardized. It's customizable.
- *It's a platform that does exactly what I want.*

Getting Started

- Get a *distribution* and install!
  - My recommendation: Ubuntu (https://www.ubuntu.com/download/desktop)
  - Use a VM if you lack spare machines :)
- Start up a Linux *virtual machine in the Cloud* and connect via SSH
  - Details on SSH in a moment
- Use the *Windows Subsystem for Linux*
  - Activate "Developer Mode" and Windows 10 / Windows Server optional feature
  - Run `bash.exe` to install Ubuntu
  - Alternatively, get other distro installers from the Microsoft Store
  - `lxrun /uninstall [/full]` gets rid of your bungled installation :)
  - Might have to `lxrun /setdefaultuser ...` or `ubuntu config --default-user ...`

The UI

```
linux:~$
```

The UI — Graphical Desktop Environments

- Gnome, KDE, Xfce, MATE, LXDE, Budgie, (Unity) ...
    - Almost endless list
    - Usually focused around a Window Manager and a design and usage philosophy
    - Often associated with a library for graphical output and system functionality (Qt, GTK)
- "Independent" Window Managers
    - Personal preference: i3

The UI — Graphical Desktop Environments

- Gnome, KDE, Xfce, MATE, LXDE, Budgie, (Unity) ...
  - Almost endless list
  - Usually focused around a Window Manager and a design and usage philosophy
  - Often associated with a library for graphical output and system functionality (Qt, GTK)
- "Independent" Window Managers
  - Personal preference: i3
- In many environments where you encounter Linux, you won't have a graphical UI!

The Shell

- Again, choices: bash, zsh, fish, ksh, tcsh, ...
- Personal preference: zsh with oh-my-zsh extensions
- Shells have several jobs:
    - Command line editing
    - Prompting
    - Execution of commands
    - Job management
    - Automation

Let's Check It Out!

The Docs

Documentation at your fingertips, since 1971.

```
$ man man
```

Remote Shells

- Command: `ssh` , for *s*ecure *sh*ell
- Connects securely to a remote system and executes a shell there
- Don't use password authentication! — Public key preferred.
- Advanced features: SSH Agent, agent forwarding, port forwarding ...
- Associated: scp
- Recommendation: check out `tmux` or `screen` for multi-pane terminal layouts
  - `tmux` , `tmux ls` `tmux attach` to work with sessions
  - `C-b d` to detach, `C-b %` and `C-b "` to create panes, `C-b right/left/up/down` to navigate
  - Reconfigure as needed!

Commands

- Shell built-ins
  - `alias` , `fg` , `cd` , `echo` , `set` , ...
- System commands in `/bin` , `/usr/bin` etc.
  - `ls` , `cp` , `mv` , `rm` , `cat` , `find` , `grep` , `less` , `ps` ...
  - On my system:

```
$ ls -l /bin | wc -l
167
$ ls -l /usr/bin | wc -l
3432
```

Command Line Features

- Aliasing: `alias`
- Completion with TAB: commands/aliases, files (wildcards — globbing)
  - Extensible completion in zsh (and others, but not so good!)
- Ctrl-R to search history
- Jobs: Ctrl-Z to suspend foreground process. `bg` (background), `fg` (foreground), `jobs`

Shell Features

- Wildcards: `*`, `**`, `?`, `[123]`, `(txt|png)`
  - Zsh extensions: `*(/)`, `*(@)`, `*(.)`
- Piping: `echo Cool thing | grep ool`
- Redirection: `grep wow /proc/* > output 2>&1`
  - `echo More content >> existingFile`
  - `cat < in > out`
- Here Documents: `cat > output <<END`
- Substitution: ``ls -l `which cat` ``
  - Or longer: `ls -l $(which cat)`

Working with the File System

- Everything is part of `/` (root)
- List: `ls`
  - Files starting with `.` are invisible, show with `ls -a`
- Directories: `mkdir`, `rmdir`, `cd`
- File operations: `cp`, `mv`, `rm`
- Create or update timestamp: `touch`
- Show file content: `cat`, `less`
- Detect file type: `file`
- Link: `ln`
- Mount: `mount`
- Space allocation: `df`, `du`
- Find files and more: `find`
- File names are *case-sensitive*!

Permissions

```
linux:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 129696 Oct  4 20:56 /bin/ls
```

Permissions

Owning group

Link count

Size

Owning user

Last Modification Timestamp

```
-rwxr-xr-x
```

Type: - (file), d (directory), c (character), ...

Owning user permissions: Read, Write, eXecute

Owning group permissions

Rest of world ("others") permissions

* A few special cases have been omitted

Manipulating Permissions

- `chown` changes owning user, `chgrp` changes owning group
  - These two usually require `sudo`, to prevent users from "giving away" their files
- `chmod` changes permissions
  - `chmod u+x` adds `x` for owning user
  - `chmod +w` adds `w` everywhere
  - `chmod 755` sets `rwxr-xr-x`
- Directories need `x` so a user can "enter" them!

User Management

- `whoami`
- `adduser` and `addgroup` create users and groups, set default shells and copy template home directories
- `usermod` modifies a user record
- `chsh` sets a user's shell
- Check `/etc/passwd` and `/etc/group` when in doubt
- `sudo` executes commands with root permissions
  - `sudo -s` gives you a root shell, but use with great caution!
  - The only known command to insult you if you want it to... google it! :)
- `su otheruser` makes you *otheruser*, `su - otheruser` uses login shell

Processes

- `ps` shows processes
  - Modern `ps` confusingly supports various option sets
  - `ps aux` and `ps -ef` show all processes, incl. lots of detail
- `top` or `htop` for prettier output and interactive features
- `kill` sends a signal to a process, `kill -l` to see signal names
  - `kill -9` is a "hard kill"
  - `skill` tries to find the process by its name

Editing Text Files

- `nano` for the ~~whimps~~ novices
  - Seriously: simple, easy to use text-based UI, recommended for first-time Linuxers
- `vi` for traditionalists
  - Very powerful, cryptic for first-time users
- `emacs` for the cool kids
  - An OS in its own right, everything-but-the-kitchensink application. Requires practice to use in text-based environments.

Configuring Things

- `/etc` is the place for system config files
  - Editing them requires `sudo`
  - It can also break your system!
  - Services may need to be restarted or instructed to reload config files, once changes have been made.
- *dot files* start with `.` (invisible, remember?) and mostly live in user home directories
  - Often they override the system-wide configuration of an application
- Many UI applications use files in `~/.config` these days

Package Management

- Package Management system depends on your distribution
  - `dpkg` , `apt` , `aptitude` on Debian/Ubuntu systems
  - `rpm` , `yum` , `dnf` , `pacman` , `emerge` , `zypper` ...
- For Ubuntu:
  - `sudo apt update` loads newest package lists
  - `sudo apt upgrade` installs available upgrades
  - `sudo apt install` and `sudo apt remove` — guess what :)
  - `do-release-upgrade` for automated upgrade to new major release versions
  - Config files in `/etc/apt`
- New: universal packages with Snap, Flatpak and AppImage

Shutting Down and other System Level Stuff

- `sudo reboot` to — ahem — reboot
- `sudo shutdown` shuts done. Duh.

Some advanced items to check out:

- `lsblk` shows connected block devices (hard drives etc.)
- `lsusb` shows USB devices
- `dmesg` shows kernel messages
- `fdisk` partitions hard drives
- In `/var/log` you can find detailed system logs
- `/boot` has files for the `grub` boot loader, and other startup items
- Learn about the `/dev` and `/proc` directories for system-level insight

Building a .NET Core App — Installing .NET Core

From https://www.microsoft.com/net/learn/get-started/linuxubuntu -

- Get Microsoft's package signature key (two lines):

```
$ curl https://packages.microsoft.com/keys/microsoft.asc |
       gpg --dearmor > microsoft.gpg
$ sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
```

- Register Microsoft's .NET Core apt package source (one line):

```
$ sudo sh -c 'echo "deb [arch=amd64]
  https://packages.microsoft.com/repos/microsoft-ubuntu-artful-prod
  artful main" > /etc/apt/sources.list.d/dotnetdev.list'
```

Building a .NET Core app — Installing .NET Core (cont.)

This is what Microsoft shows on the web page:

```
$ sudo apt-get install apt-transport-https
$ sudo apt-get update
$ sudo apt-get install dotnet-sdk-2.1.4
```

The highlighted line may be required in certain "limited functionality" environments, but not on normally installed workplace machines.

Building a .NET Core app — Installing VS Code

From https://code.visualstudio.com/docs/setup/linux :

- Get Microsoft's package signature key — you have already done this
- Register the VS Code apt package source (one line):

```
$ sudo sh -c 'echo "deb [arch=amd64]
  https://packages.microsoft.com/repos/vscode stable main" >
  /etc/apt/sources.list.d/vscode.list'
```

- And install:

```
$ sudo apt update
$ sudo apt install code
```

Creating a .NET Core MVC App

Simples!

```
$ mkdir demoapp
$ cd demoapp
$ dotnet new mvc
$ dotnet run
```

Deployment Considerations

- Physical deployment
  - `dotnet publish`
  - Copy to `/usr/local/APPNAME` — recommended location!
- Running the app in Kestrel is easy and fast
  - The standard template call `CreateDefaultBuilder` uses Kestrel
- However, Microsoft recommends against running Kestrel as a public front-end server
- Use a reverse proxy!
  - Recommendation: nginx
  - Apache or IIS (or others) also possible
- Plan: Configure Kestrel and nginx for automatic startup of the web app.

Setting Up nginx

```
$ sudo apt install nginx
$ cd /etc/nginx
$ sudo vi sites-available/demoapp
$ sudo ln -s sites-available/demoapp sites-enabled/
$ sudo nginx -s reload
```

demoapp config:

```
server {
  listen 80;
  location / {
    proxy_pass http://localhost:5000;
  }
}
```

Setting Up nginx

```
$ sudo apt install nginx
$ cd /etc/nginx
$ sudo vi sites-available/demoapp
$ sudo ln -s sites-available/demoapp sites-enabled/
$ sudo nginx -s reload
```

Watch Out

In the nginx default setup, there may be a server configuration listening on port 80 already. If so, you can deactivate it by removing the symbolic link:

```
$ sudo rm /etc/nginx/sites-enabled/default
```

Setting Up nginx

```
$ sudo apt install nginx
$
$
$
$
```

den

```
s

}
```

Watch Out

In the nginx default setup, there may be a server

deactivate it by removing the symbolic link:

```
$ sudo rm /etc/nginx/sites-enabled/default
```

Note that this Setup is not Complete!

Microsoft has full instructions for nginx (and others) here:

https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?tabs=aspnetcore2x

Auto-Starting Kestrel

```
$ sudo vi /etc/systemd/system/demoapp.service
$ sudo systemctl enable demoapp
$ sudo systemctl start demoapp
```

Auto-Starting Kestrel — demoapp.service

```
[Unit]
Description=Demo App

[Service]
WorkingDirectory=PROJECTDIR
ExecStart=/usr/bin/dotnet PROJECTDIR/ ... /demoapp.dll
Restart=always
RestartSec=10
SyslogIdentifier=demoapp
User=www-data
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

Auto-Starting Kestrel — demoapp.service

```
[Unit]
Description=Demo App

[Service]
WorkingDirectory=PROJECTDIR
ExecStart=/usr/bin/dotnet PROJECTDIR/.../demoapp.dll
Restart=always
RestartSec=10
SyslogIdentifier=demoapp
User=[illegible]
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

There's More to Say

Like before, this setup is not the end of the story. Find Microsoft's full instructions here:

https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?tabs=aspnetcore2x

Sources

- This presentation:
  - https://oliversturm.github.io/linux-101-for-dotnet-devs
  - PDF download: https://oliversturm.github.io/linux-101-for-dotnet-devs/slides.pdf

Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers • oliver@oliversturm.com

DevExpress®