

# Production d'un système embarqué

**Christophe BLAESS**

[christophe.blaess@logilin.fr](mailto:christophe.blaess@logilin.fr)

<https://www.linkedin.com/in/christophe-blaess/>

<https://www.blaess.fr/christophe/>

twitter: @chrisblaess



Ingénierie et formations sur Linux et les logiciels libres

<https://www.logilin.fr>

<b>Découverte du système Linux embarqué.....</b>	<b>3</b>
Arborescence typique.....	3
Travaux pratiques : contenu de l'arborescence des fichiers.....	4
Travaux pratiques : éléments produits par Buildroot.....	5
<b>Configuration du système.....</b>	<b>6</b>
Nom d'hôte et message de connexion.....	6
Travaux pratiques : personnalisation, configuration des utilisateurs.....	8
Protection du système de fichiers.....	9
Répertoire d'overlay de l'arborescence.....	10
Travaux pratiques : ajout de scripts de remontage du rootfs.....	11
Scripts de démarrage.....	12
<b>Ajout de commandes et d'applications.....</b>	<b>13</b>
Busybox, couteau suisse de l'embarqué.....	13
Configuration de Busybox.....	14
Travaux pratiques : configuration des applets de Busybox.....	15
Ajout d'applications dans Buildroot.....	16
Travaux pratiques : ajout d'outils de base du système.....	17
<b>Configuration du réseau.....</b>	<b>18</b>
Configuration automatique du réseau avec DHCP.....	18
Configuration statique du réseau.....	19
Travaux pratiques : mise en œuvre d'un service ssh.....	20
Serveur HTTP.....	22
Synchronisation d'horloge.....	23

<b>Noyau Linux.....</b>	<b>24</b>
Les licences libres.....	25
Drivers spécifiques.....	27
Travaux pratiques : configuration du noyau Linux.....	28

Ce support de formation est distribué sous licence **Creative Commons 4.0**



*(Attribution - Partage dans les mêmes conditions).*

Vous êtes libres de copier et partager ce document, en mentionnant son origine. Si vous l'intégrez dans un contenu plus vaste, ce dernier devra être distribué avec les mêmes droits.

Ce cours a été rédigé en utilisant des logiciels libres sur système d'exploitation Linux :

- *LibreOffice Writer* pour le support et la mise en page
- *LibreOffice Draw* pour les dessins vectoriels
- *Gimp* pour les images bitmap

Linux embarqué

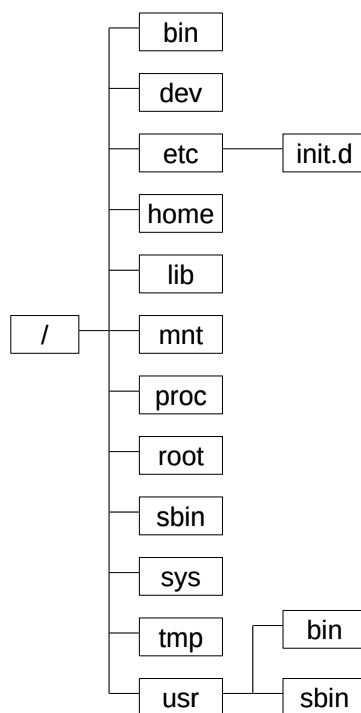
ILE v.5.4

<https://www.blaess.fr/christophe/>

<https://www.logilin.fr>

# Découverte du système Linux embarqué

## Arborescence typique



Le système de fichiers virtuel `devtmpfs` permet de remplir automatiquement `/dev` (sur un disque RAM). Le noyau monte automatiquement ce système de fichiers dès le démarrage.

Le système de fichiers virtuel `proc` exporte dans `/proc` des informations du noyau concernant les processus et les paramètres du *kernel*.

Le système de fichiers virtuel `sysfs` exporte dans `/sys` la vision du noyau sur le système matériel et les périphériques.

## Travaux pratiques : contenu de l'arborescence des fichiers

### Structure et contenu de l'arborescence de fichiers

*Authentifiez-vous et essayez quelques commandes usuelles :*

```
Welcome to Buildroot
buildroot login: root
```

```
# cd /
```

```
# ls
```

```
bin          lib          media        root          tmp
dev          lib32        mnt          run           usr
etc          linuxrc      opt          sbin          var
home         lost+found  proc         sys
```

```
# uname -a
```

```
Linux buildroot 6.1.44 #1 SMP Sat Oct 05 09:07:30 CEST 2024 armv7l
GNU/Linux
```

```
# cat /proc/cpuinfo
```

```
model name   : ARMv7 Processor rev 0 (v7l)
[...]
Hardware     : ARM-Versatile Express
```

```
# free
```

	total	used	free	shared	buff/cache	available
Mem:	249612	12000	236032	44	1580	235276
Swap:	0	0	0			

Le message « *random: nonblocking pool is initialized* » qui peut apparaître au bout de quelques dizaines de secondes indique simplement que Linux a terminé l'initialisation de son générateur de nombres aléatoires.

## Travaux pratiques : éléments produits par Buildroot

Identifiez les éléments suivants dans le résultat de Buildroot :

- **Kernel** (noyau du système)
- **Device Tree** (descriptif du matériel)
- **Busybox** (exécutable implémentant l'essentiel des commandes présentes)
- **LibC** (bibliothèque assurant l'interface entre espace utilisateur et kernel)

### Solutions :

*Kernel* : `images/zImage` à partir du répertoire de build sur la machine hôte

*Device Tree* : `images/vexpress-v2p-ca9.dtb` sur la machine hôte

*Busybox* : `/bin/busybox` sur la cible

*LibC* : `/lib/libuClibc-1.xx.xx.so` sur la cible

# Configuration du système

## Nom d'hôte et message de connexion

### Personnalisation du nom d'hôte

Le nom d'hôte et le message de bienvenue sur le système sont configurables dans le menu « *System Configuration* » de Buildroot

```
Buildroot 2015.02 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> will exclude a feature. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature

Target options --->
Build options --->
Toolchain ---->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->

<Select> < Exit > < Help > < Save > < Load >
```

```
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> will exclude a feature. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature

[Buildroot] System hostname
(Welcome to Buildroot) system banner
Passwords encoding (nds) --->
Init system (BusyBox) --->
/dev management (Dynamic using devtmpfs only) --->
(system/device_table.txt) Path to the permission tables
Root FS skeleton (default target skeleton) --->
(root) Root password
/bin/sh (busybox' default shell) --->
[*] Run a getty (login prompt) after boot
    getty options --->
[ ] remount root filesystem read-write during boot
    L(+)

<Select> < Exit > < Help > < Save > < Load >
```

## ***Utilisateurs et mots de passe***

Le mot de passe de *root* est configurable dans le menu « *System Configuration* »

La liste des autres utilisateurs est fournie à Buildroot dans une table au format suivant :

<login> <uid> <groupe> <gid> <password> <home> <shell> <groupes supp.> <nom>

...

## Travaux pratiques : personnalisation, configuration des utilisateurs

*Créez un fichier `users.tbl` dans un répertoire de configuration, contenant :*

```
guest -1 users -1 =guest /home/guest /bin/sh - Embedded System User
```

*Indiquez le chemin dans l'option « Path to the users tables » du menu « System configuration ».*

*Par exemple :*

**`$(TOPDIR)/../config/users.tbl`**

*Il faut ensuite re-générer l'image et relancer l'émulateur.*



## Protection du système de fichiers

Il est préférable, pour des raisons de robustesse, de laisser le système de fichier principal (*rootfs*) en lecture seulement.

Un système de fichiers en lecture-écriture est susceptible de contenir des (petites) incohérences en cas de coupure brutale d'alimentation électrique.

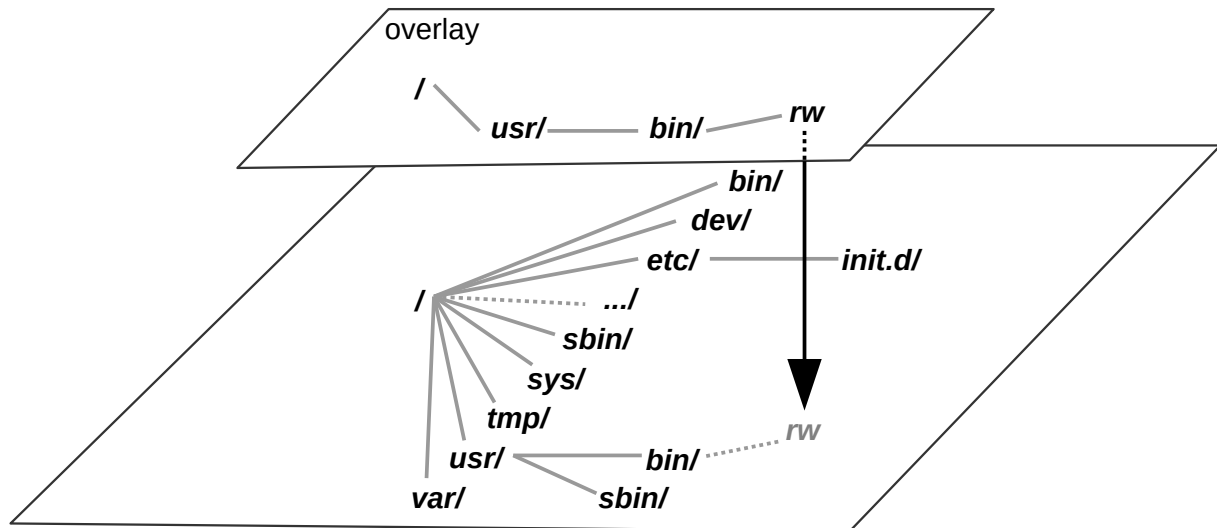
Or le *rootfs* doit être monté par le *kernel* durant le *boot*, et celui-ci ne peut pas gérer ces incohérences. Il n'a pas accès aux outils comme *fsck* (*file system check*) car ils se trouvent dans l'arborescence.

Conserver un *rootfs* en lecture seule est une bonne pratique de l'embarqué.

Dans le menu « *System configuration* » de Buildroot, désactivez l'option « *remount root filesystem read-write during boot* ».

## Répertoire d'overlay de l'arborescence

L'option « *Root filesystem overlay directories* » du menu « *System configuration* » peut contenir un répertoire de base qui sera superposé à l'arborescence du système de fichiers de la cible.



## Travaux pratiques : ajout de scripts de remontage du rootfs

Créez et ajoutez un *overlay* dans votre configuration de Buildroot, contenant deux scripts shells :

```
/usr/bin/rw:  
#!/bin/sh  
mount / -o rw,remount
```

et

```
/usr/bin/ro:  
#!/bin/sh  
mount / -o ro,remount
```

PS : n'oubliez pas de rendre les scripts shell exécutables...

## Scripts de démarrage

Le processus `init` lit le fichier `/etc/inittab` et agit en conséquence.

Ce fichier contient une action à réaliser par ligne.

*Chaque ligne contient plusieurs champs :*

`<Terminal>:<Niveau d'exécution (ignoré)>:<Type d'action>:<Commande>`

*Le type d'action est généralement `sysinit` ou `respawn`.*

*Par exemple, la ligne suivante initialise le système de fichiers `/proc` :*

```
::sysinit:/bin/mount -t proc proc /proc
```

*Et la ligne suivante lance les scripts de démarrage :*

```
::sysinit:/etc/init.d/rcS
```

Le script `rcS` lance successivement tous les scripts se trouvant dans `/etc/init.d/` dont le nom commence par un `S` suivi d'au-moins deux caractères :

```
# ls /etc/init.d/  
S01syslogd  S02sysctl  S40network  rcS  
S02klogd    S20urandom  rcK
```

Le script `S40network` démarre des services réseau étudiés plus loin.

# Ajout de commandes et d'applications

## Busybox, couteau suisse de l'embarqué

L'utilitaire **Busybox** implémente plus de 400 commandes courantes. En voici quelques-unes :

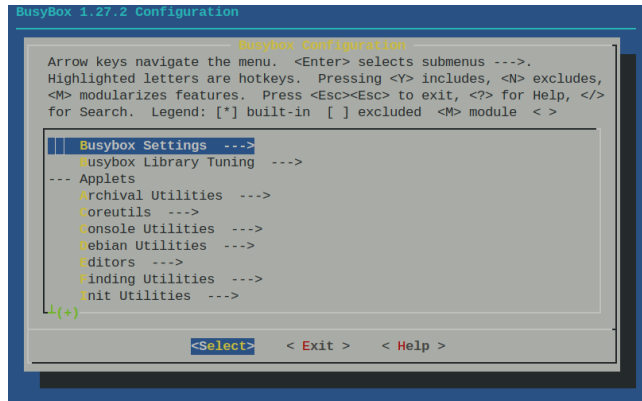
```
[, [[, addgroup, adduser, adjtimex, ar, arp, arping, ash, awk, base64, basename,
beep, blkid, brctl, bunzip2, bzip2, cal, cat, catv, chatr, chgrp, chmod,
chown, chroot, chrt, chvt, cksum, clear, cmp, cp, cpio, crond, crontab, cryptpw,
cut, date, dc, dd, deallocvt, delgroup, deluser, depmod, devmem, df, dhcprelay,
diff, dirname, dmesg, dnsd, dnsdomainname, dos2unix, du, dumpkmap, dumpleases,
echo, ed, egrep, eject, env, expr, false, fatattr, fbset, fbsplash, fdflush,
fdformat, fdisk, fgrep, find, flash_eraseall, flashcp, fold, free, freeramdisk,
fsck, fstrim, ftpd, ftpget, ftpput, fuser, getopt, getty, grep, groups, gunzip,
gzip, halt, head, hexdump, hostid, hostname, httpd, hwclock, id, ifconfig, ifdown,
ifup, inetd, init, insmod, iostat, ip, kill, killall, klogd, last, less, ln,
loadfont, loadkmap, logger, login, logname, losetup, ls, lsattr, lsmmod, lsof,
lspci, lsusb, lzcat, md5sum, mesg, microcom, mkdir, mkdosfs, mke2fs, mkfifo,
mkfs.ext2, mkfs.vfat, mknod, mktemp, modinfo, modprobe, more, mount, mountpoint,
mpstat, mt, mv, nameif, nanddump, nandwrite, nc, netstat, nice, nmeter, nohup,
nslookup, ntpd, od, openvt, passwd, patch, pgrep, pidof, ping, pipe_progress,
pivot_root, pkill, pmap, poweroff, powertop, printenv, printf, ps, pscan, pstree,
pwd, rdate, rdev, readlink, readprofile, realpath, reboot, renice, reset, resize,
rev, rm, rmdir, rmmod, route, rtcwake, run-parts, runlevel, rx, script,
scriptreplay, sed, seq, setarch, setconsole, setkeycodes, setlogcons, setserial,
setsid, sh, sha1sum, sha256sum, sha3sum, sha512sum, sleep, sort, start-stop-daemon,
stat, strings, stty, su, sulogin, sum, sync, sysctl, syslogd, tac, tail, tar,
taskset, tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, top, touch, tr,
traceroute, true, tty, tune2fs, udhcp, udhcpd, udpsvd, umount, uname, uncompress,
unexpand, uniq, unix2dos, unlink, unzip, uptime, usleep, uudecode, uuencode, vi,
vlock, wall, watch, watchdog, wc, wget, which, who, whoami, xargs, yes, zcat
```

Buildroot configure, compile, et intègre automatiquement Busybox avec les scripts et fichiers de configuration nécessaires.

## Configuration de Busybox

Le choix des commandes embarquées dans Busybox s'effectue dans un menu de configuration spécifique,

On peut (dé)sélectionner des « applets » de Busybox et pour certaines d'entre-elles, configurer des options de comportement.



Avec Buildroot, ce menu s'obtient en appelant :

**\$ make busybox-menuconfig**

## Travaux pratiques : configuration des applets de Busybox

Ajoutez dans Busybox les applets suivantes :

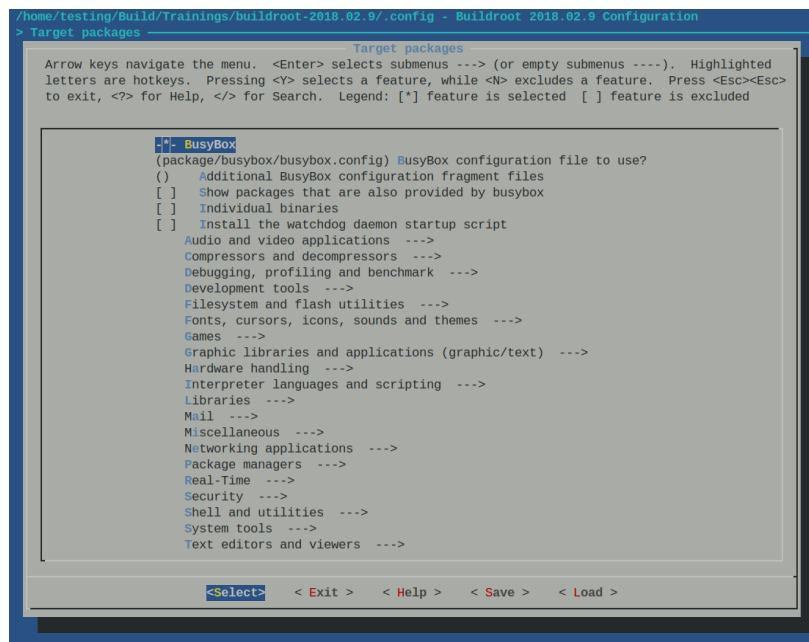
```
Linux System Utilities --->  
  [*] cal  
  [*] taskset
```

```
Networking Utilities --->  
  [*] httpd  
  [*] ntpd
```

Recompilez et relancez le système, vérifiez la présence des utilitaires souhaités.

## Ajout d'applications dans Buildroot

Le menu « *Target packages* » de Buildroot contient plus de 4000 options concernant environ 2000 packages différents !



Certaines applications ne peuvent être compilées que si des options spécifiques (principalement de la *toolchain*) sont activées.



## Travaux pratiques : ajout d'outils de base du système

Ajoutez les applicatifs suivants (et d'autres si vous le souhaitez) dans la configuration de Buildroot :

Debugging, profiling, benchmark --->  
[\*] gdb

Networking applications --->  
[\*] dropbear  
[\*] disable reverse DNS lookups

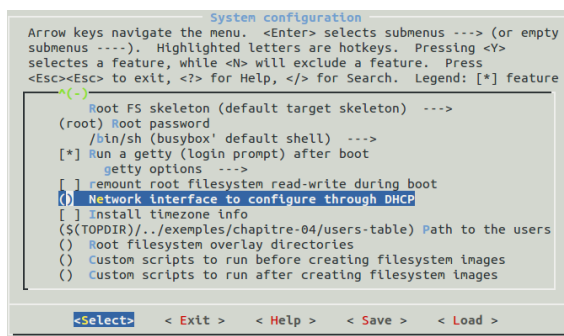
Text editors and viewers --->  
[\*] nano

Recompilez, réinstallez le système et vérifiez la présence des nouveaux outils.

# Configuration du réseau

## Configuration automatique du réseau avec DHCP

Buildroot peut préparer directement les fichiers pour qu'une interface réseau soit configurée automatiquement avec le protocole DHCP.



```
System configuration
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty
submenus ---). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> will exclude a feature. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature
[*]
  Root FS skeleton (default target skeleton) --->
  (root) Root password
  /bin/sh (busybox' default shell) --->
  [*] Run a getty (login prompt) after boot
  getty options --->
  [ ] remount root filesystem read-write during boot
  (N) Network interface to configure through DHCP
  [ ] Install timezone info
  ($TOPDIR)/../examples/chapitre-04/users-table Path to the users
  ( ) Root filesystem overlay directories
  ( ) Custom scripts to run before creating filesystem images
  ( ) Custom scripts to run after creating filesystem images

<Select> < Exit > < Help > < Save > < Load >
```

## Configuration statique du réseau

Ajoutez dans votre overlay un répertoire **etc/network/**

Créez dans ce répertoire un fichier **interfaces** avec l'aspect suivant

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.1.200    ← à ajuster en fonction de
    gateway 192.168.1.10   ← votre environnement réseau
    netmask 255.255.255.0
```

Vous pouvez utiliser comme point de départ le fichier :

build-qemuarm/target/etc/network/interfaces

Créez un fichier **etc/resolv.conf** dans l'arborescence overlay contenant

```
nameserver 8.8.8.8
```

Vous pouvez remplacer le DNS de Google (8.8.8.8) indiqué plus haut par celui de votre Fournisseur d'Accès Internet, ou par un DNS ouvert comme 80.67.169.12 (*French Data Network*).

## Travaux pratiques : mise en œuvre d'un service ssh

Le serveur **Dropbear** offre une connexion sécurisée de bout en bout grâce au protocole SSH.

Notre système de fichiers étant en lecture seulement, les clés ne sont pas persistantes lors du redémarrage. Nous pouvons y remédier.

*Supprimer le répertoire dropbear temporaire (lien vers /var)*

```
# rw
# rm /etc/dropbear
# mkdir /etc/dropbear
```

*Générer une clé aléatoire :*

```
# dropbearkey -t ecdsa -f /etc/dropbear/dropbear_ecdsa_host_key
Generating 256 bit ed25519 key, this may take a while...
Public key portion is:
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIA4ywhx0Nlnra8z3/t20C6GLiEm
B5bW9thp1Kb130fum root@Qemu
Fingerprint: sha1!!
53:20:08:21:ee:ef:16:24:56:4b:d1:77:d8:1f:99:3c:33:b0:c0:08
```

*Réinitialiser le service :*

```
# ro
# reboot
```

*Sur le PC : essayer une connexion normale*

```
$ ssh root@192.168.1.200
The authenticity of host '192.168.1.200 (192.168.1.200)' can't be
established.
ecdsa key fingerprint is b8:66:6c:f8:90:44:21:59:71:e9:f1:a5:6b:4b:f4:a0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.200' (ECDSA) to the list of known
hosts.
root@192.168.1.200's password: ***
# rw
# exit
```

*Sur le PC : échanger les clés pour une connexion automatique*

```
$ ssh-copy-id root@192.168.1.200
root@192.168.1.200's password: ***
```

*Sur le PC : envoyer un fichier local vers la cible :*

```
$ echo "Hello World!" > message.txt
$ scp message.txt root@192.168.1.200:/tmp/
```

*Sur le PC : relire le fichier message après modification sur la cible :*

```
$ scp root@192.168.1.200:/tmp/message.txt .
$ cat message.txt
```

## Serveur HTTP

Busybox inclut un petit serveur HTTP supportant l'invocation de scripts CGI.

*Sur la cible, créez un répertoire pour les fichiers HTML :*

```
# mkdir /home/www/
```

*Créez un fichier HTML minimal, par exemple :*

/home/www/index.html :

```
<HTML>
  <BODY>
    <H1>Hello</H1>
    <p>
      Hello from this embedded server
    </p>
  </BODY>
</HTML>
```

*Sur la cible, lancez le serveur HTTP :*

```
# httpd -h /home/www
```

*Accédez avec le navigateur de votre PC à l'adresse de la cible  
par exemple : <http://192.168.1.200/>*

## Synchronisation d'horloge

Lorsque l'horloge n'est pas maintenue (absence d'horloge RTC sauvegardée par une pile), il faut se synchroniser sur une base de temps lors du boot. On emploie souvent NTP (*Network Time Protocol*).

Busybox inclut un démon `ntpd`.

```
# date
Thu Jan  1 00:00:47 UTC 1970
# ntpd -d -n -q -p pool.ntp.org
ntpd: 'pool.ntp.org' is 85.199.214.98
ntpd: sending query to 85.199.214.98
ntpd: reply from 85.199.214.98: offset:+1633762483.863927 delay:0.014323
ntpd: sending query to 85.199.214.98
ntpd: reply from 85.199.214.98: offset:+1633762483.864307 delay:0.015328
ntpd: setting time to 2021-10-09 06:55:43.083548 (offset +1633762483.864
# date
Sat Oct 05 10:02:15 CEST 2024
#
```

*Si une horloge RTC sauvegardée est présente on peut la mettre à jour avec :*

```
# hwclock -w
```

*la lire avec :*

```
# hwclock -r
```

*et mettre l'heure système à jour avec :*

```
# hwclock -s
```

# Noyau Linux

Le **noyau** Linux représente le cœur du système d'exploitation, il répartit les ressources matérielles (temps processeur, mémoire, périphériques...) entre les applications et permet de disposer d'un système :

- **multi-tâches** : temps-partagé préemptif classique, temps-réel souple ;
- **multi-utilisateur** : modèle Unix avec utilisateurs et administrateur *root* ;
- **multi-processeur** : support des multiprocesseurs et multicœurs ;
- **multi-architectures** : x86 32/64, PPC, Arm, Alpha, Mips, RiscV, etc.

Les sources du noyau Linux sont disponibles simultanément en plusieurs versions :

- versions *stable* : par exemple 5.17.2
- versions *longterm* : par exemple 5.15.33
- version *release candidate* : par exemple 5.18-rc2
- version *development* : par exemple next-20220411

Il y a une nouvelle version stable du noyau tous les deux mois environ (nouveaux drivers, protocoles, systèmes de fichiers, améliorations globales, etc.)

A titre indicatif, le code source du noyau Linux 5.0 contient :

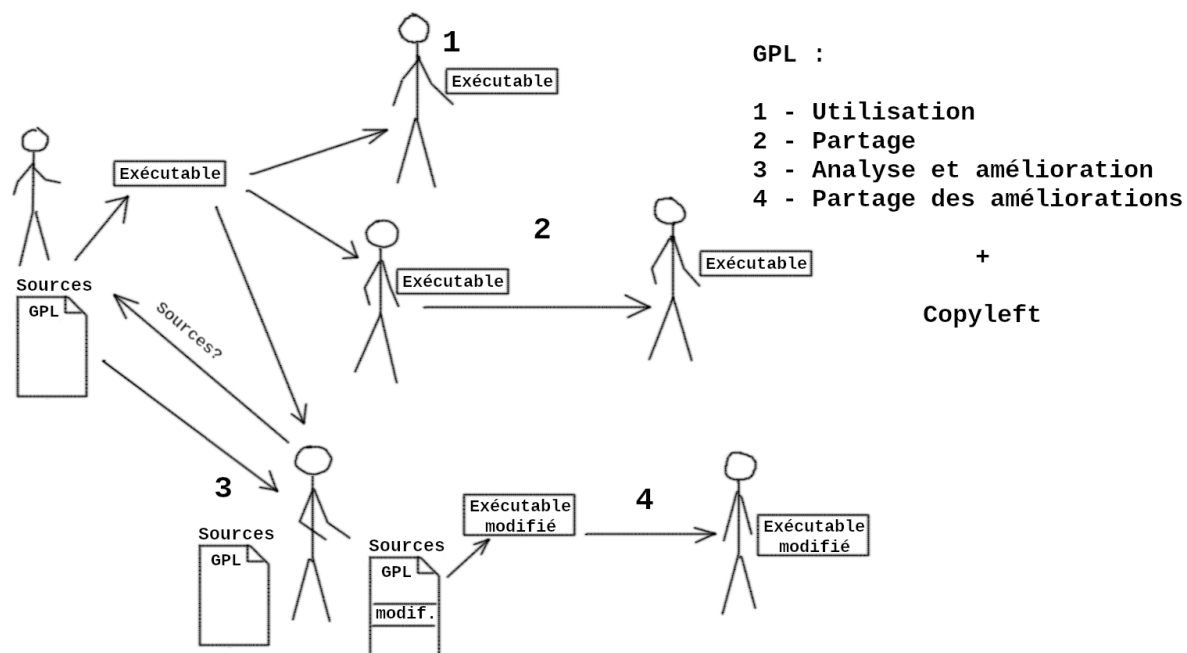
- 934 Mo de code source (dont 549 Mo pour le sous-système *drivers*/)
- 63.090 fichiers (dont 25.218 dans *drivers*/)
- 26.632 fichiers de code source C (dont 15.650 dans *drivers*/)
- 19.317 fichiers d'en-tête C (.h)
- 1.324 fichiers assembleurs (.S)
- 2.455 fichiers de compilation Makefile
- 6314 fichiers de documentation
- 19.042.339 lignes de code source
- 16.400 options de compilation

## Voir aussi

- Les sources du noyau : <http://www.kernel.org/>
- Actualités hebdomadaires sur le développement noyau : <http://lwn.net/kernel/>



## Les licences libres



Un logiciel sous licence GPL v.3 (et v.3 uniquement) ne doit pas être employé dans un environnement qui empêche l'utilisateur de le recompiler et remplacer la version distribuée.

### Voir aussi

- Point de vue de l'OSI : <http://opensource.org/licenses>
- Point de vue de la FSF : <http://www.gnu.org/licenses/>

## ***Licence du noyau Linux***

Le noyau Linux est sous licence GPLv2.

Les modules développés pour le noyau Linux doivent contenir une description de leur licence

<i>GPL</i>	GPL v. 2 ou ultérieure
<i>GPLv2</i>	GPL v. 2
<i>GPL and additional rights</i>	Utilisé pour l'intégration de drivers initialement sous licences plus permissives que la GPL.
<i>Dual BSD/GPL</i>	Redistribution au choix sous les conditions de la GPL v.2 ou selon les critères des licences BSD, MIT, MPL respectivement
<i>Dual MIT/GPL</i>	
<i>Dual MPL/GPL</i>	
<i>Proprietary</i>	Module propriétaire

Un module propriétaire ne peut pas être lié statiquement dans le kernel.

Les modules propriétaires sont tolérés si chargés après le boot.

Du code sous licence MPL peut être intégré (sans aucune modification) dans du code propriétaire.

La licence BSD ne contient pas la clause de *copyleft*. Le code sous licence BSD est libre, mais peut être modifié et intégré dans du code propriétaire.

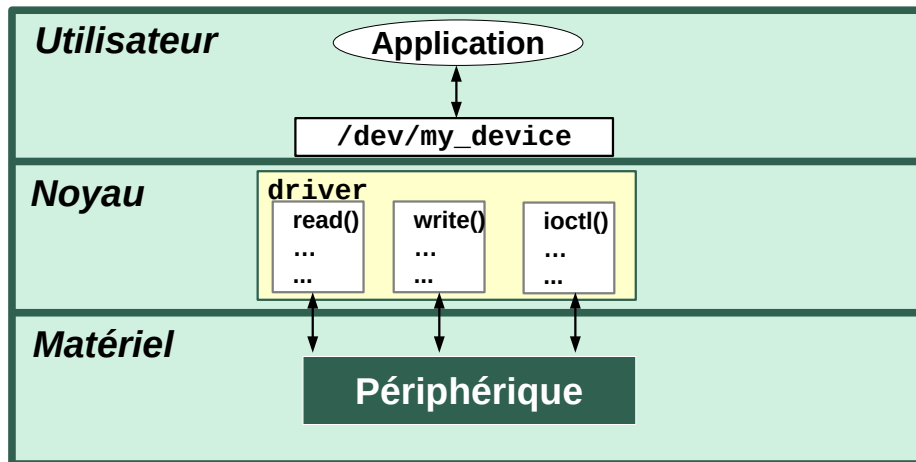
### **Pour en savoir plus...**

[JEAN] Benjamin Jean : « *Option Libre, Du bon usage des licences libres* » – Framasoft.  
<https://framabook.org/optionlibre-dubonusagedeslicenceslibres/>,

## Drivers spécifiques

Le support pour les périphériques est assuré par des pilotes (drivers) qui peuvent être développés de manière externe au noyau (modules).

Les applications de l'espace utilisateur communiquent avec les périphériques en réalisant des opérations (lecture, écriture paramétrage...) sur des pseudo-fichiers (généralement stockés dans /dev)



La compatibilité de l'API interne du kernel n'est pas garantie entre deux versions stables successives (ex. 4.17 et 4.18), même si les changements sont généralement minimes.

[MADIEU] : John Madiou – *Linux Device Drivers Development*.

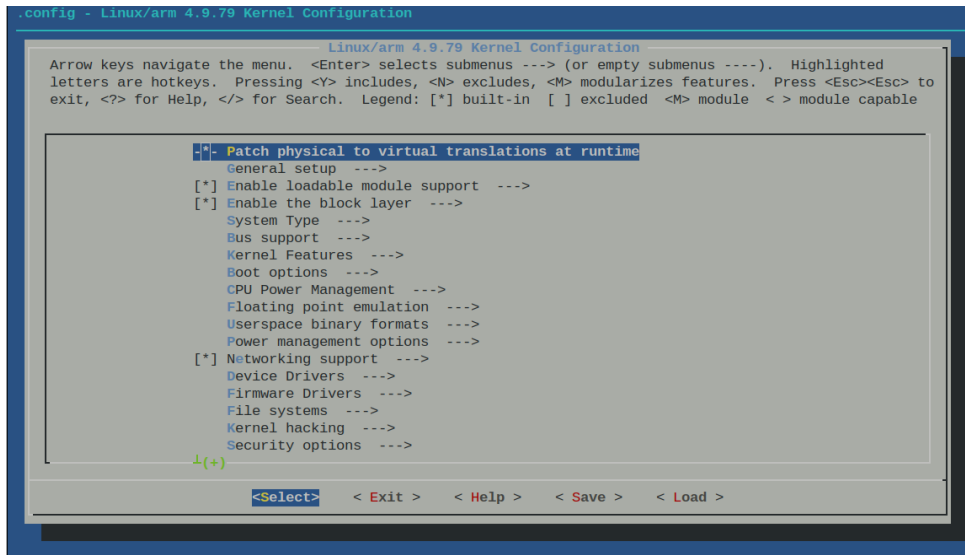
Le développement de drivers robustes et portables entre différentes versions du noyau n'est pas une opération triviale (surtout sur des architectures multi-cœurs) !

Conseil, support, formation et expertise : <https://www.logilin.fr>

## Travaux pratiques : configuration du noyau Linux

Avec Buildroot, la configuration du noyau Linux s'effectue dans un menu spécifique que l'on obtient avec la commande :

```
$ make linux-menuconfig
```



Il y a plus de 15 000 options de compilations (avec des dépendances) !

Éditez la configuration du noyau Linux.

Dans le menu :    `General setup --->`

Éditez la ligne :

`(-v7) Local version - append to kernel release`

Pour effacer ce qui suit le tiret (v7) et ajouter un nom de votre choix (par exemple vos initiales) :

`(-custom) Local version - append to kernel release`

Dans le menu :    `Kernel Features --->`

Vous éditez le sous-menu :    `Preemption Model`

Pour sélectionner l'option :

`(X) Preemptible Kernel (Low-Latency Desktop)`

Sauvez la configuration, recompilez et réinstallez le système.

Vérifiez le résultat avec la commande : `uname -a`