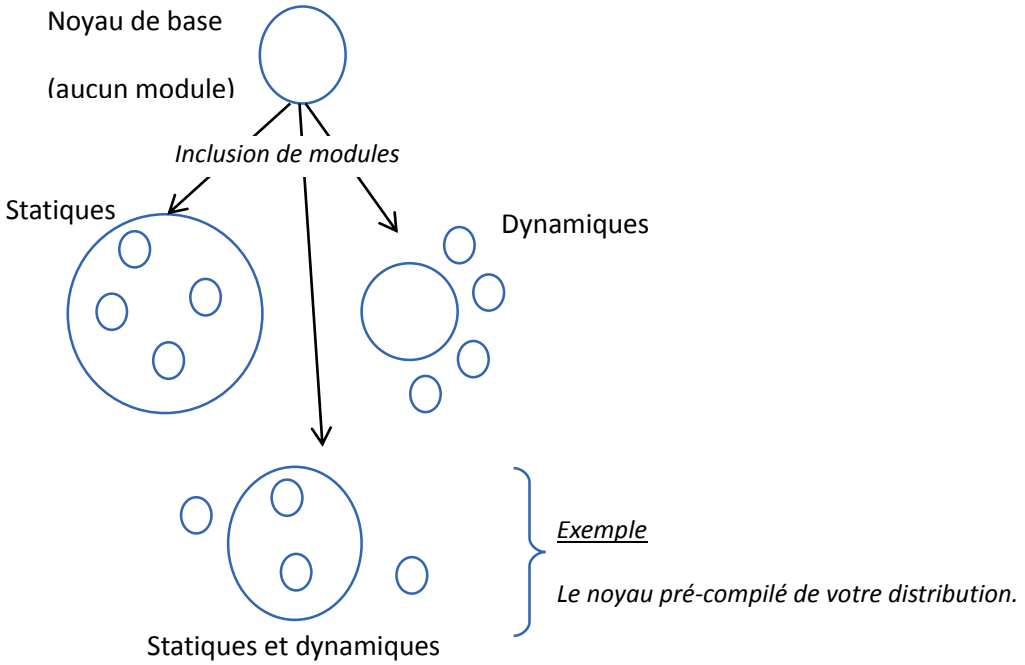
Le noyau: Configuration

- L'étape la plus importante avant de compiler un noyau Linux est sa configuration.
  On peut choisir globalement d'avoir un noyau statique ou dynamique.

  Concrètement, on va faire le choix des modules qui composent le noyau.
  On choisit aussi si l'inclusion du support est statique ou sous forme d'un noyau chargé dynamiquement
  La configuration permet aussi également la prise en compte ou non d'un module ou d'un sous-système.



  L'atelier suivant vous initie à cette étape de configuration.

- Les outils nécessaires pour cet atelier:

  `kernel-devel` -> package qui contient les outils requis (les Makefile, kernel-headers, l'arborescence des sources du noyau …) pour pouvoir compiler des modules externes et assurer la configuration du noyau. **Il ne contient pas les sources** de celui-ci.

  `make` -> commande qui permet d'exécuter une série d'ordres se trouvant dans un fichier Makefile.
  (http://fr.openclassrooms.com/informatique/cours/compilez-sous-gnu-linux)

  `ncurses-devel` -> une bibliothèque de fonctions fournissant une API pour le développement d'environnements en mode texte. Utilitaires basés sur cette bibliothèque:

  Midnight Commander, make menuconfig …

  `gcc` -> compilateur C utilisé pour l'exécution de commande lancées par make

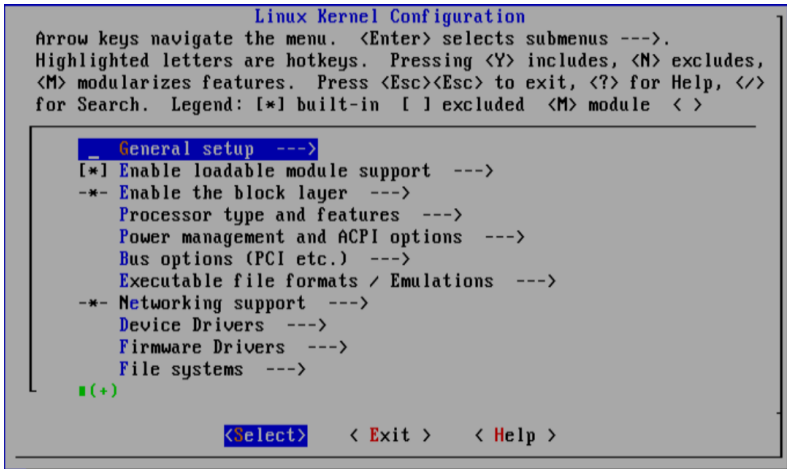
Atelier 1

1. Installez si nécessaire les packages ci-dessus #yum install kernel-devel make ncurses-devel gcc -y

2. Placez-vous dans /usr/src/kernels/$(uname –r) -> arborescence créée après l'installation de kernel-devel
   #cd /usr/src/kernels/2.6[...]
   a. Un fichier Makefile doit s'y trouver

   b. Un fichier .config doit s'y trouver (contient une configuration par défaut du noyau)

   c. Entrez dans le menu de configuration du noyau #make menuconfig



   d. On navigue dans les menus avec les flèches, on valide ses choix grâce à "Select"

      M Le support sera compilé sous forme de module
      Y Le support sera compilé sous forme statique
      N Le support ne sera pas pris en compte

      Ajouter la gestion du file system reiserfs en module
      File systems --> Reiserfs module --> M
      Vérification de l'architecture du processeur
      Processor type and features
      Vérification de la prise en compte des architecture SMP et NUMA
      Processor type and features --> SMP Processor type and features --> NUMA
      On configure un noyau pouvant gérer maximum 4 processeurs

   e. On termine en sortant des différents menus par "Exit" et on enregistre le fichier de configuration (Choix "Yes").

   f. On vérifie que nos choix ont été bien pris en compte

Modules du noyau

- Certaines fonctionnalités du noyau Linux peuvent être compilées indépendamment de celui-ci. Elles sont compilées sous forme de module. L'utilisation des modules tend à se généraliser.

  Avantages:
  - réduction de la taille du noyau car les modules non-utilisés depuis un certain temps sont déchargés automatiquement de la Ram;
  - configuration dynamique du matériel ;
  - possibilité d'ajout de certaines fonctionnalités sans recompiler le noyau ;
  - des pilotes de périphériques peuvent être livrés sous forme de binaires;

  Inconvénients: complexité accrue.

- Les commandes:

`lsmod`	->	pour afficher la liste de ceux qui sont actuellement chargés par le noyau
`rmmod`	->	pour supprimer un module chargé
`insmod`	->	pour charger un module (le chemin d'accès complet au module être indiqué)
`modprobe`	->	pour charger un module ainsi que ses dépendances (le chemin d'accès complet au module ne doit pas être indiqué)
`depmod`	->	crée le fichier qui mémorise les dépendances entre modules
`modinfo`	->	affiche les informations concernant un module

- Les fichiers:

  `/lib/modules/$(uname -r)` -> arborescence où sont stockés les modules (extention *.ko*) d'une version du noyau

  `/lib/modules/$(uname -r)/modules.dep` -> contient toutes les dépendances entre les modules

  `/etc/modprobe.d/*.conf` -> fichiers de configuration des modules

Atelier 2

1. Quel est le module qui gère votre carte réseau ?
   e1000
2. L'accès au réseau est-il opérationnel ?
   Oui
3. Ce module dépend-t-il d'un autre module ?
   Non
4. Déchargez ce module de la Ram.
   #modprobe -r e1000
   a. Vérifiez s'il est bien enlevé.
      #lsmod | grep e1000
   b. L'accès au réseau est-il opérationnel ?
      Non
5. Rechargez ce module de la Ram.
   #modprobe -a e1000
   a. Vérifiez s'il est bien rechargé.
      #lsmod | grep e1000
   b. L'accès au réseau est-il opérationnel ?
      Oui
6. Déchargez ce module de la Ram et déplacez le module physique dans /tmp

#modprobe -r e1000 #mv /lib/modules/2.6.32-696.e16.x86_64/kernel/drivers/net/e1000/e1000.ko /tmp

7. Rechargez ce module de la Ram.
   #modprobe -a e1000
   a. Que se passe-t-il ?

WARNING: Could not open /lib/modules/2.6.32-696.e16.x86_64/kernel/drivers/net/e1000/e1000.ko No such file or directory

8. Rétablir une situation fonctionnelle.

#mv /tmp/e1000.ko /lib/modules/2.6.32-696.e16.x86_64/kernel/drivers/net/e1000/ #modprobe -a e1000

9. Afficher des informations sur ce module
   #modinfo e1000

Atelier 3

1. Je désire supprimer le module 'fat'. Comment procéder ?
   #rmmod fat
2. Comment rétablir la situation
   a. à l'aide de la commande modprobe ?
      #modprobe -a fat
   b. à l'aide de la commande insmod ?
      #insmod /lib/modules/2.6.32-696.e16.x86_64/kernel/fs/fat/fat.ko

Atelier 4

1. Quels sont les FS pris en charge actuellement par le noyau (soit de manière statique soit par un module actuellement chargé) ?
   #cat /proc/filesystems | cut -f2 | pr -4t
2. Quels sont les FS chargeables par des modules ?
   #ls /lib/modules/2.6.32-696.e16.x86_64/kernel/fs/

*Sur les 2 points précédents, on constate qu'il n'y a pas de module pour **sysfs** (il est inclus dans le noyau de manière statique). Inversément, actuellement **cifs** n'est pas pris en charge mais on peut charger le module correspondant.*

3. Affichez la configuration des modules (celle utilisée lors de leur chargement automatique)
   PAS VU
4. Forcez le chargement du module cifs au démarrage

echo "modprobe cifs" > /etc/rc.modules #chmod +x /etc/rc.modules #reboot #lsmod | grep cifs

5. Comment les supports gérant les FS btrfs et iso9660 ont-ils été compilés en sachant que le fichier **/boot/config-$(uname –r)** contient toutes les options de compilation du noyau ?
   #cat /boot/config -2.6... |grep btrfs (PAS SUR)

Atelier 5

Comment connaître le contenu de l'initramfs ?

PAS VU

Remarque : La commande 'dracut' permet de (re)générer un initramfs...

COMMANDES ET FICHIERS MANIPULES LORS DE CES ATELIERS

Nouvelles commandes:

lsmod – rmmod – insmod – modprobe – ping – make menuconfig – file – zcat - cpio

Fichiers et dossiers:

Makefile - .config

/lib/modules/$(uname –r)/
/lib/modules/$(uname –r)/modules.dep
/lib/modules/$(uname -r)/kernel/drivers/net/e1000/e1000.ko
/lib/modules/$(uname –r)/kernel/fs/
/lib/modules/$(uname -r)/kernel/fs/fat/fat.ko
/lib/modules/$(uname -r)/kernel/fs/fat/vat.ko

/proc/filesystems

/etc/modprobe.d/
/etc/modprobe.d/*.conf
/etc/modprobe.d/dist.conf

/etc/sysconfig/modules/

/boot/config-$(uname -r)

Packages:

kernel-devel - make - ncurses-devel - gcc