TD : taille maximale d'un fichier au format **ext2**

Le format **ext2**

Dans ce TD, nous nous intéressons au format obsolète **ext2** dont voici le schéma des champs d'un inode :

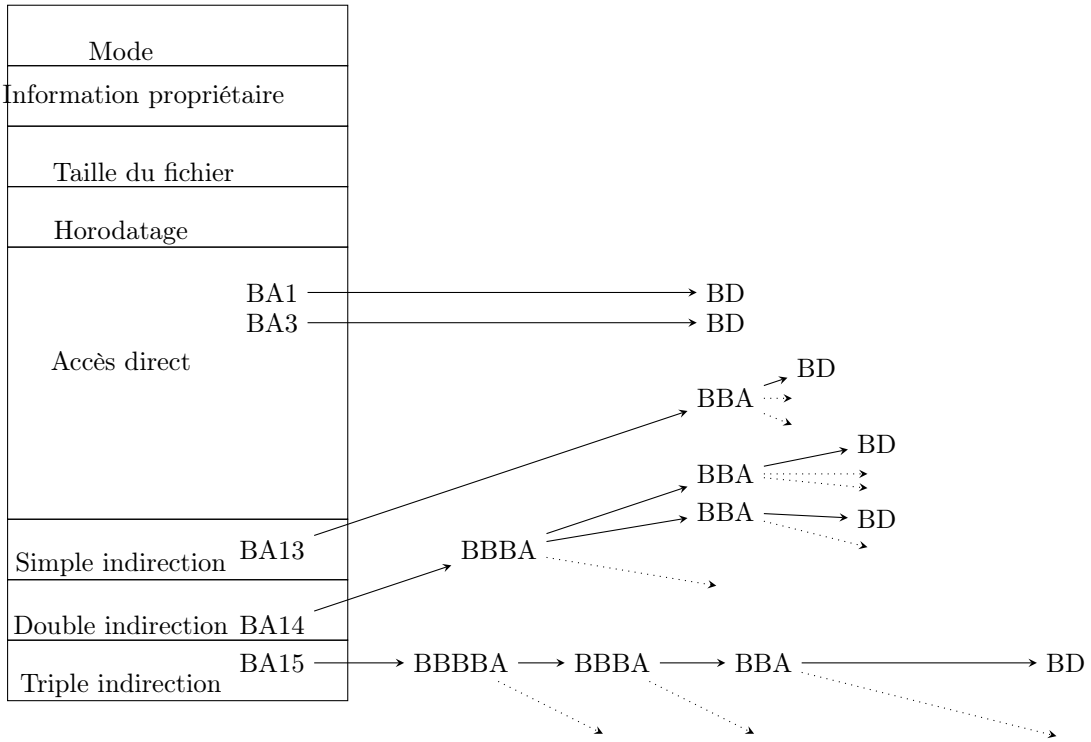

FIGURE 1 – Les champs du format de fichier **ext2**. **BD** : bloc de données ; **BA** : bloc d'adresse ; **BBA** : bloc de données qui contient des blocs d'adresse ; **BBBA** : bloc de données qui contient des blocs d'adresses qui pointent vers des blocs de données qui contiennent des blocs d'adresse etc.

Rappelons qu'un inode contient toutes les informations sur un fichier donné à l'exception de son nom (lequel est indiqué dans le répertoire parent du fichier). Voici quelques explications vus en cours à propos des attributs de l'inode :

1. Le champ **mode** regroupe deux sortes d'information :

— la nature de l'objet décrit : fichier régulier, répertoire, lien symbolique, périphérique bloc, périphérique caractère et ou encore pipe, sockets...
— les permissions de ce fichier.

2. Le champ **Informations propriétaire** identifie le propriétaire du fichier ou du répertoire.
3. Le champ **taille** se passe de commentaire.
4. Le champ **Horodatage** contient la date de création, la date de dernière modification des données et celle de la dernière modification de l'inode (ce sont deux dates différentes puisqu'on peut changer les droits sans toucher aux données).
5. Les blocs de données. Ce sont en fait des pointeurs vers les blocs (zones du disque) qui contiennent effectivement les données.

   Citons la documentation du noyau linux pour le format **ext2** :

   « There are pointers to the first 12 blocks which contain the file's data in the inode. There is a pointer to an indirect block (which contains pointers to the next set of blocks), a pointer to a doubly indirect block and a pointer to a trebly indirect block »

   — Les 12 premiers champs sur les 15 contiennent les adresses des 12 premiers blocs de données.
   — Si ce n'est pas suffisant pour stocker toute l'information du fichier, on utilise les champs 13,14, 15 dans l'ordre.
   Le bloc de simple indirection (bloc 13) est un pointeur sur un bloc de données qui contient des pointeurs qui pointent sur des blocs de données.
   Le bloc de double indirection (bloc 14) est un pointeur qui pointe vers un bloc de données qui contient des pointeurs vers des blocs de données qui contiennent des pointeurs vers des blocs de données.
   Le bloc 15 est associé à une triple indirection.

   Remarque. Comme la taille des systèmes de fichiers est grande, le risque de perdre des données augmente lors d'un arrêt brutal du disque. Le principe des *systèmes de fichiers journalisés* est de conserver les données de gestion dans un journal (*log*) plutôt que sur le disque. Cela assure une meilleure fiabilité au système de fichiers en cas d'arrêt brutal du serveur. Au démarrage, le système de fichier est vérifié et réparé grâces aux données contenues dans le journal.

   Les systèmes de fichiers **ext3** et **ext4** sont des variantes journalisées de **ext2**.

   En cas de gros fichier, on remplit d'abord les blocs de données directement accessibles, puis ceux accessible par une indirection à un niveau etc.

Exercice

Dans cet exercice, on suppose que les blocs de données ont des tailles de 1024 octets et que la taille des blocs d'adresses est de 32 bits. On suppose en outre que tous les blocs de données (accessibles directement ou non) sont pleins.

Q1 Quelle est la tailles en octet des données accessibles par les 12 premiers blocs ?

Q2 Combien de blocs d'adresses peut contenir un bloc de données ?

Q3 Vers combien d'adresse pointe le bloc 13 ? Combien cela représente-t-il de blocs de données ? Quelle est la place occupée par les blocs de données accessibles en une indirection ?

Q4 Combien de blocs de données sont accessibles en deux indirections à partir du bloc 14 ? Quelle place est occupée ?

Q5 Quelle place est occupée par les données accessibles en 3 indirections depuis le bloc 15 ?

Q6 Conclure en calculant la taille du fichier de taille maximale pour le format **ext2**. Donner cette taille en octets, Gio (gibioctet) puis Go (gigaoctet).