

# Implantation des objets externes sur disque

# Linéarisation de l'espace

- Position d'un secteur

nc 0..nbc - 1 numéro de cylindre

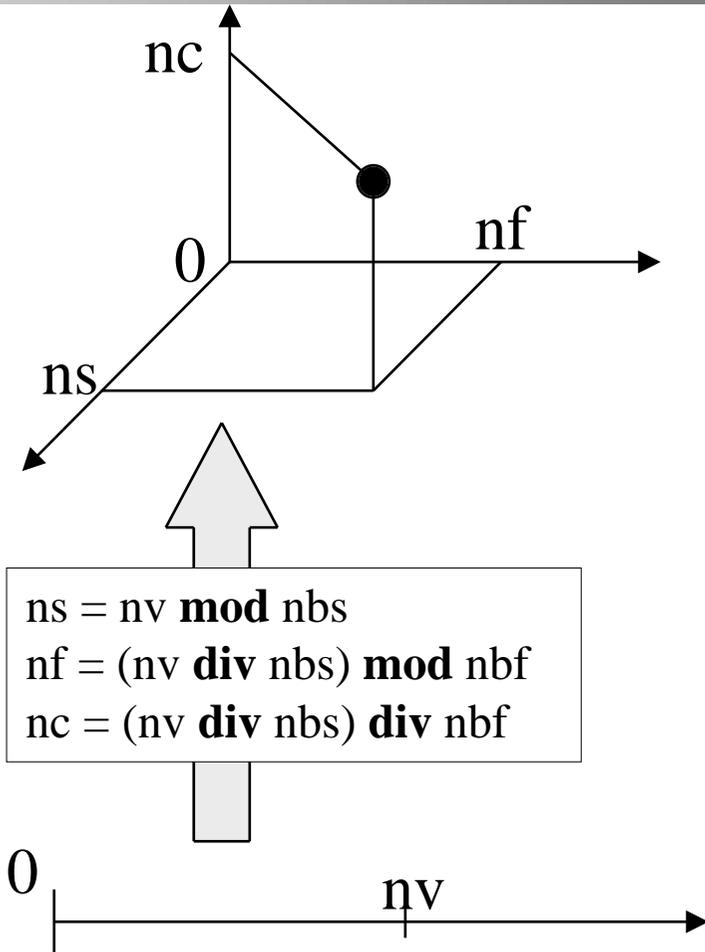
nf 0..nbf - 1 numéro de face

ns 0..nbs - 1 numéro de secteur

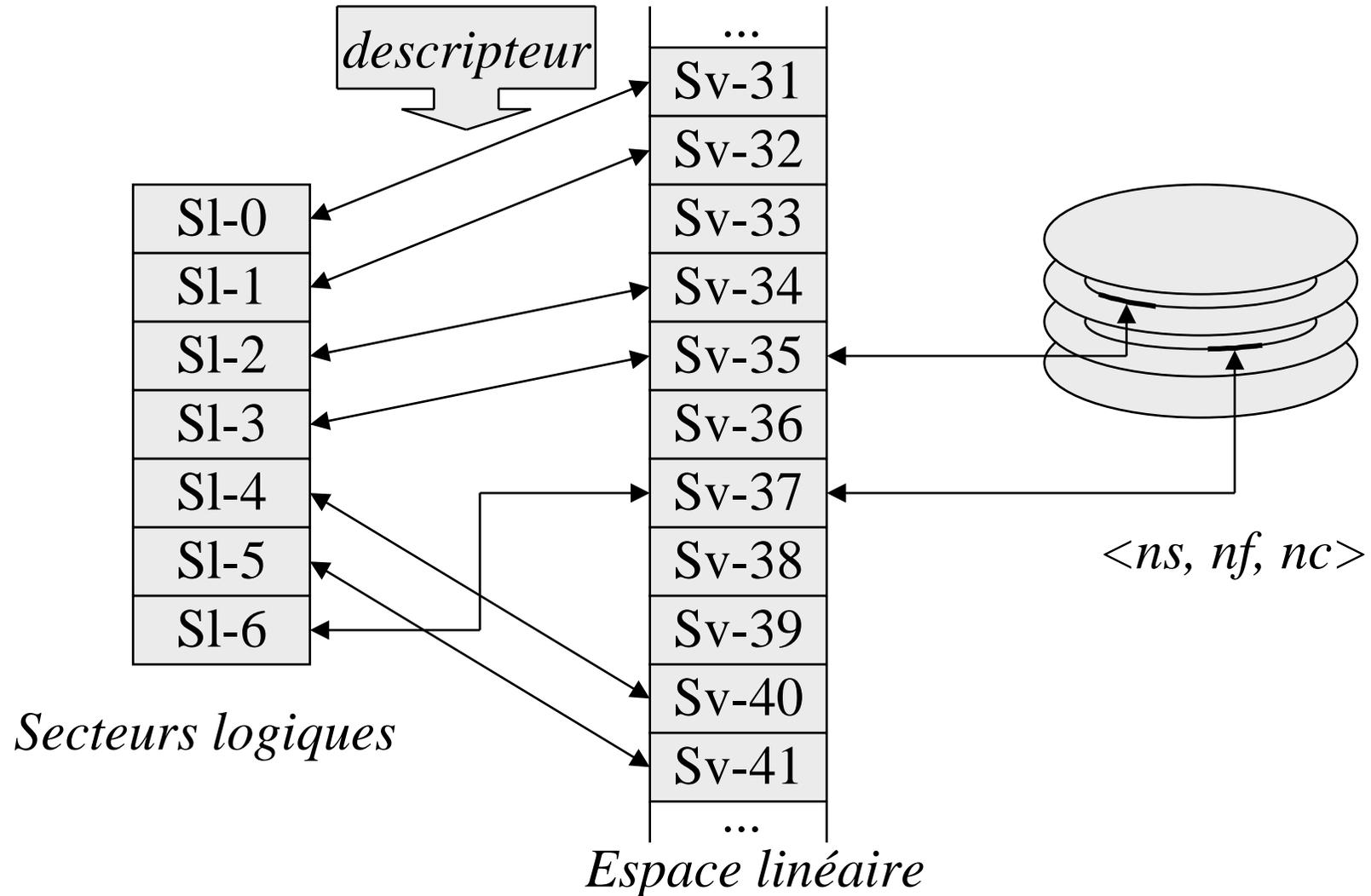
$$nv = ns + nbs * (nf + nbf * nc)$$

nv 0..N - 1 numéro virtuel

où  $N = nbs * nbf * nbc$



# Notion de descripteur



# Allocation par zone (1)

- Implantation simple

espace contigu dans l'espace linéaire

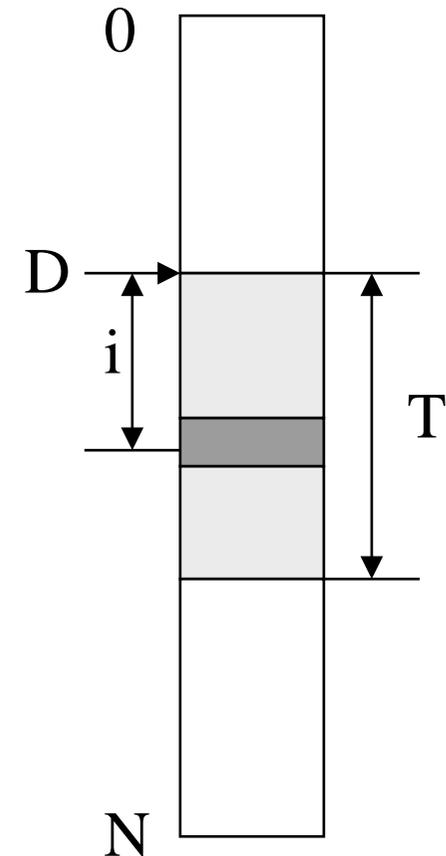
début en  $D$ , taille  $T$

secteur  $i$  de l'objet  $\rightarrow nv = D + i \rightarrow \langle ns, nf, nc \rangle$

avantage: simplicité, peu de mouvements de bras

inconvénients

- difficulté pour agrandir
- connaissance de taille exacte à la création
- impossibilité de trouver  $T$  secteurs consécutifs libre, alors que le total des libres  $> T$



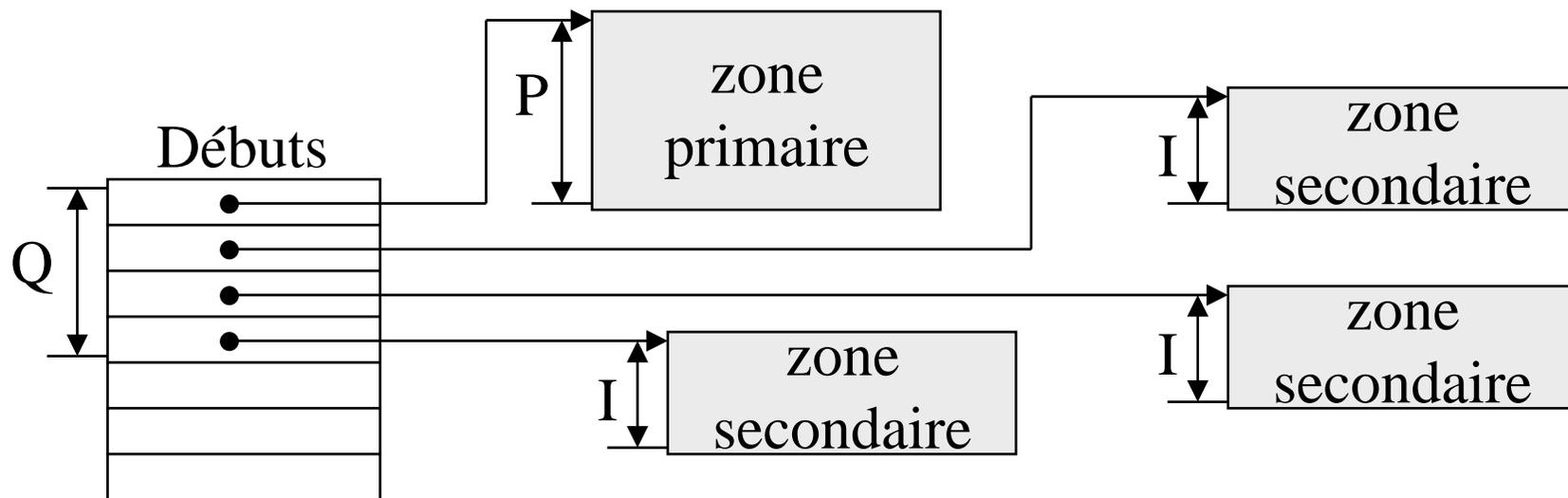
# Allocation par zone (2)

- Implantations avec extensions fixes

taille commune à toutes des extensions:  $\langle P, I \rangle$

secteur  $i$  ( $\langle P \rangle \rightarrow nv = \text{Débuts}[0] + i \rightarrow \langle ns, nf, nc \rangle$

(  $P \rangle \rightarrow nv = \text{Débuts}[(i-P)\text{div } I + 1] + (i-P)\text{mod } I \rightarrow \langle ns, nf, nc \rangle$

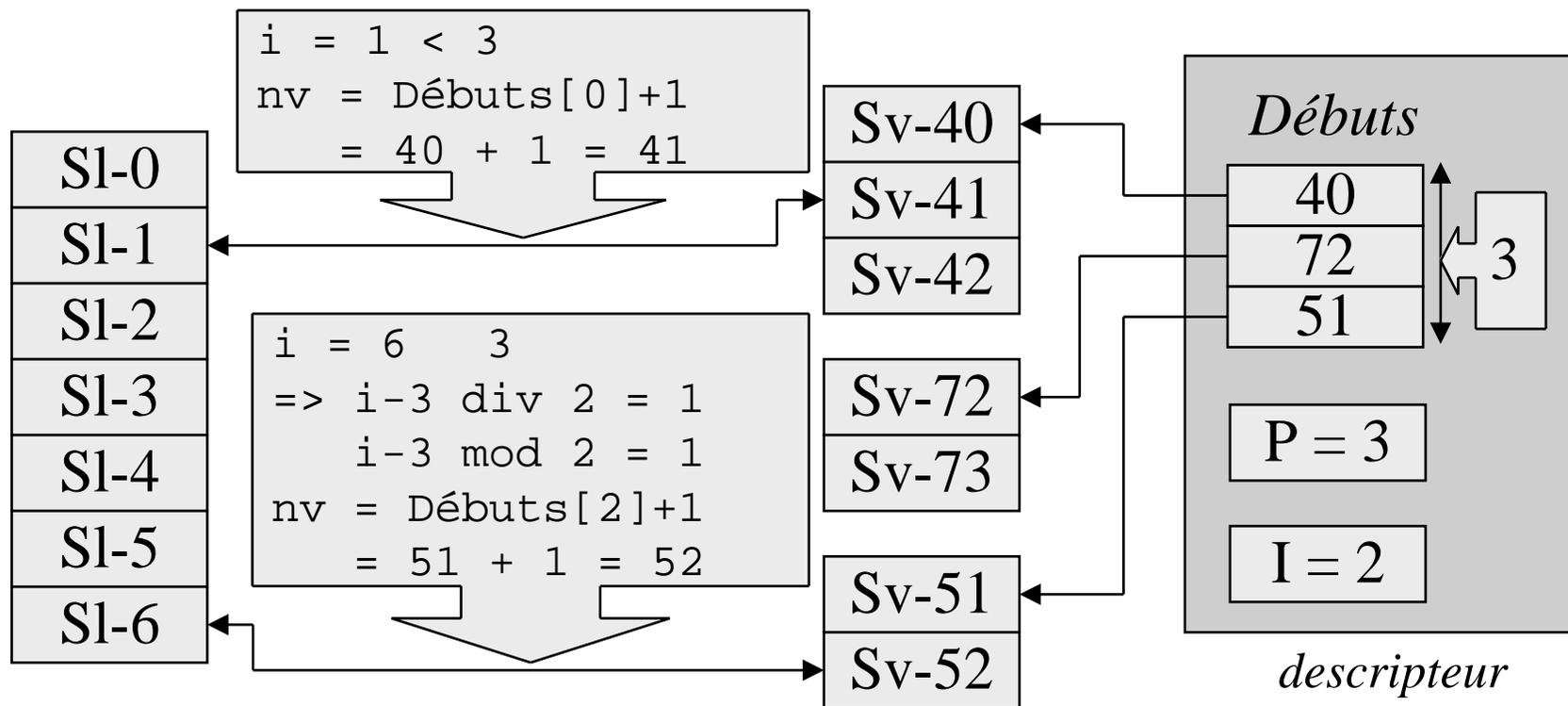


# Allocation par zone (3)

- Exemple,  $P = 3$  et  $I = 2$

secteur  $i$  ( $<P$ )  $\rightarrow$   $nv = \text{Débuts}[0] + i \rightarrow \langle ns, nf, nc \rangle$

(  $P$ )  $\rightarrow$   $nv = \text{Débuts}[(i-P)\text{div } I + 1] + (i-P)\text{mod } I \rightarrow \langle ns, nf, nc \rangle$



# Allocation par zone (4)

taille effective:  $T = P + (Q - 1) * I$

taille maximum:  $T = P + (M - 1) * I$ , où M nombre d'entrées de Début

besoin réel  $R \Rightarrow$  perte  $T - R = \max(P - R, I - 1)$

perte moyenne minimale si:

$$I = \frac{(R_{\max} - R_{\min} + 1)}{M} \quad \text{où } R_{\min} \leq R \leq R_{\max}$$

$$P = R_{\max} - (M - 1) * I$$

si P trop grand (pas de zone contiguë de cette taille), on peut prendre une valeur telle que:

$$P = \frac{R_{\max}}{M}$$

conclusions

- légère perte d'efficacité si plusieurs zones
- possibilité d'allonger dans certaines limites
- impossibilité éventuelle de créer est réduite, mais existe encore

# Allocation par zone (5)

- Implantations avec extensions quelconque

nombre quelconque d'espaces contigus, de taille quelconque

$\langle \langle D_1, T_1 \rangle, \langle D_2, T_2 \rangle, \dots, \langle D_n, T_n \rangle \rangle$

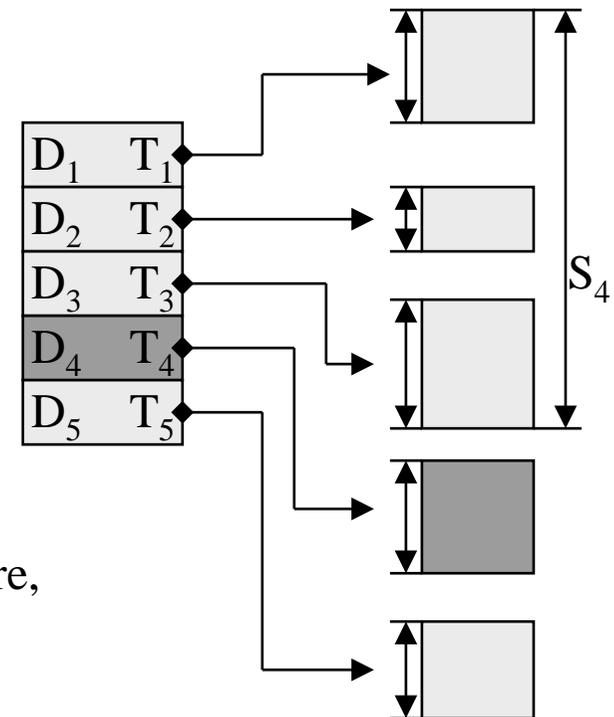
secteur  $i \rightarrow nv = D_j + i - S_j \rightarrow \langle ns, nf, nc \rangle$

où  $S_j = \sum_{k=1}^{j-1} T_k$  et  $j$  tel que  $S_j \leq i < S_{j+1}$

en parcours séquentiel, calcul de  $S_j$  au fur et à mesure,  
en accès aléatoire, boucle de calcul

si disque morcelé, allocation possible, mais  
fragmentation de l'objet externe

exemples: MacOS, NTFS et VMS



# Allocation par bloc de taille fixe (1)

- Bloc = cluster = nombre fixe de secteurs (nbsb)

blocs numérotés,  $j \rightarrow nv_j = nbsb * j \{ + constante \}$

- Implantation par blocs chaînés (FAT)

chaînage des blocs d'un objet externe dans la FAT

secteur  $i$  de l'objet  $\rightarrow r := i \text{ div } nbsb;$

$j := D;$  { premier bloc de l'objet }

**tant que**  $r > 0$  **faire**

$j := FAT[j];$

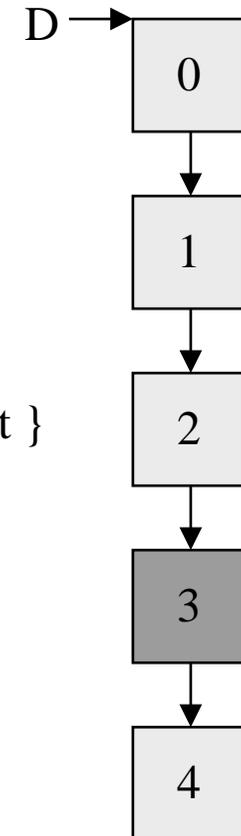
$r := r - 1;$

**fait;**

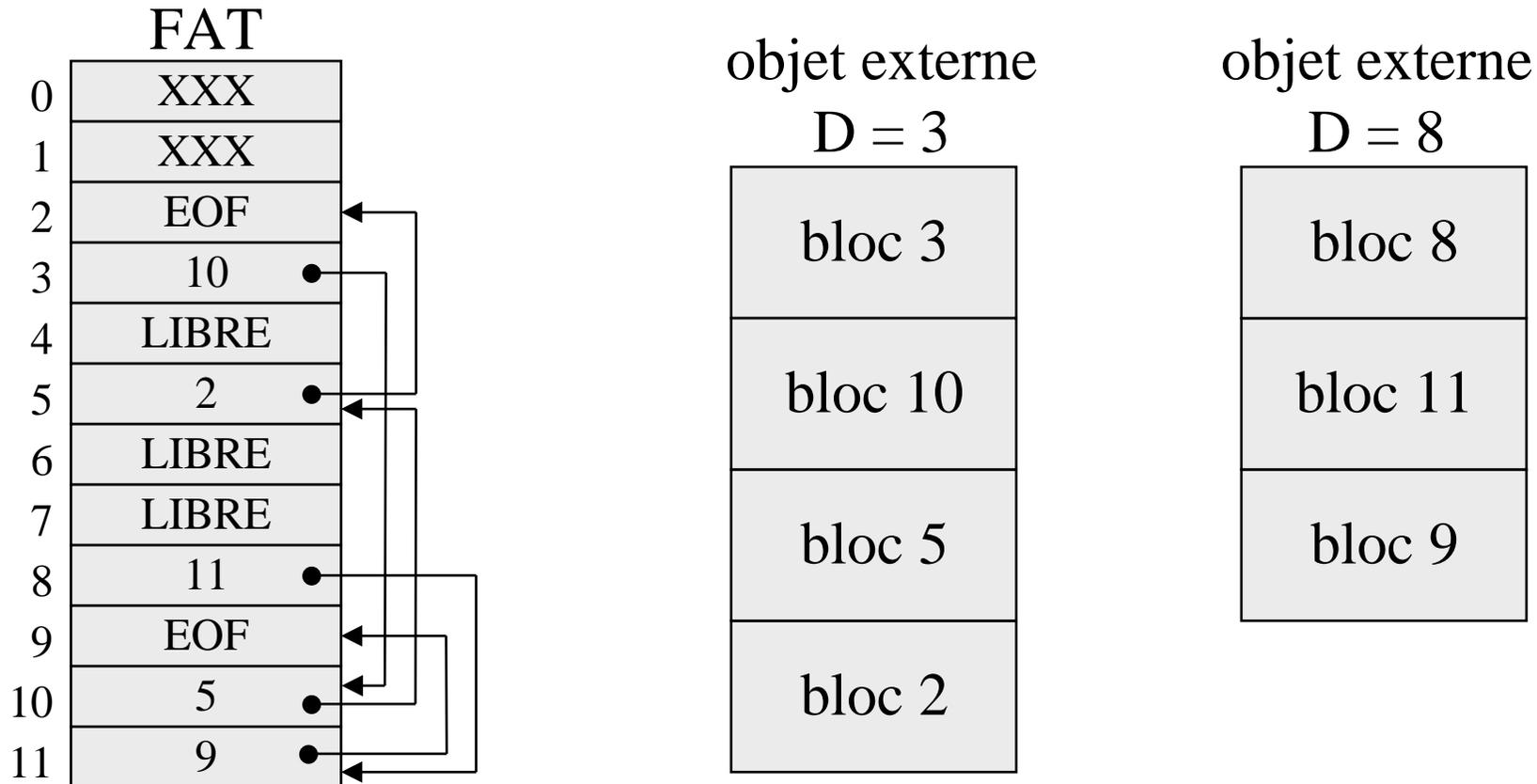
$nv = nv_j + i \text{ mod } nbsb; \rightarrow \langle ns, nf, nc \rangle$

convient bien aux objets liés aux fichiers séquentiels

pour les fichiers aléatoires, calcul proportionnel à  $r$



# Allocation par bloc de taille fixe (2)



problème: taille de la FAT pour les disques de grandes tailles  
par exemple 32 Mo, et blocs de 1 Ko => 64 Ko  
si non en mémoire centrale => allongement des temps d'accès

# Allocation par bloc de taille fixe (3)

- Implantation par blocs à plusieurs niveaux

pour chaque objet  $\Leftrightarrow$  une table de description de l'espace alloué

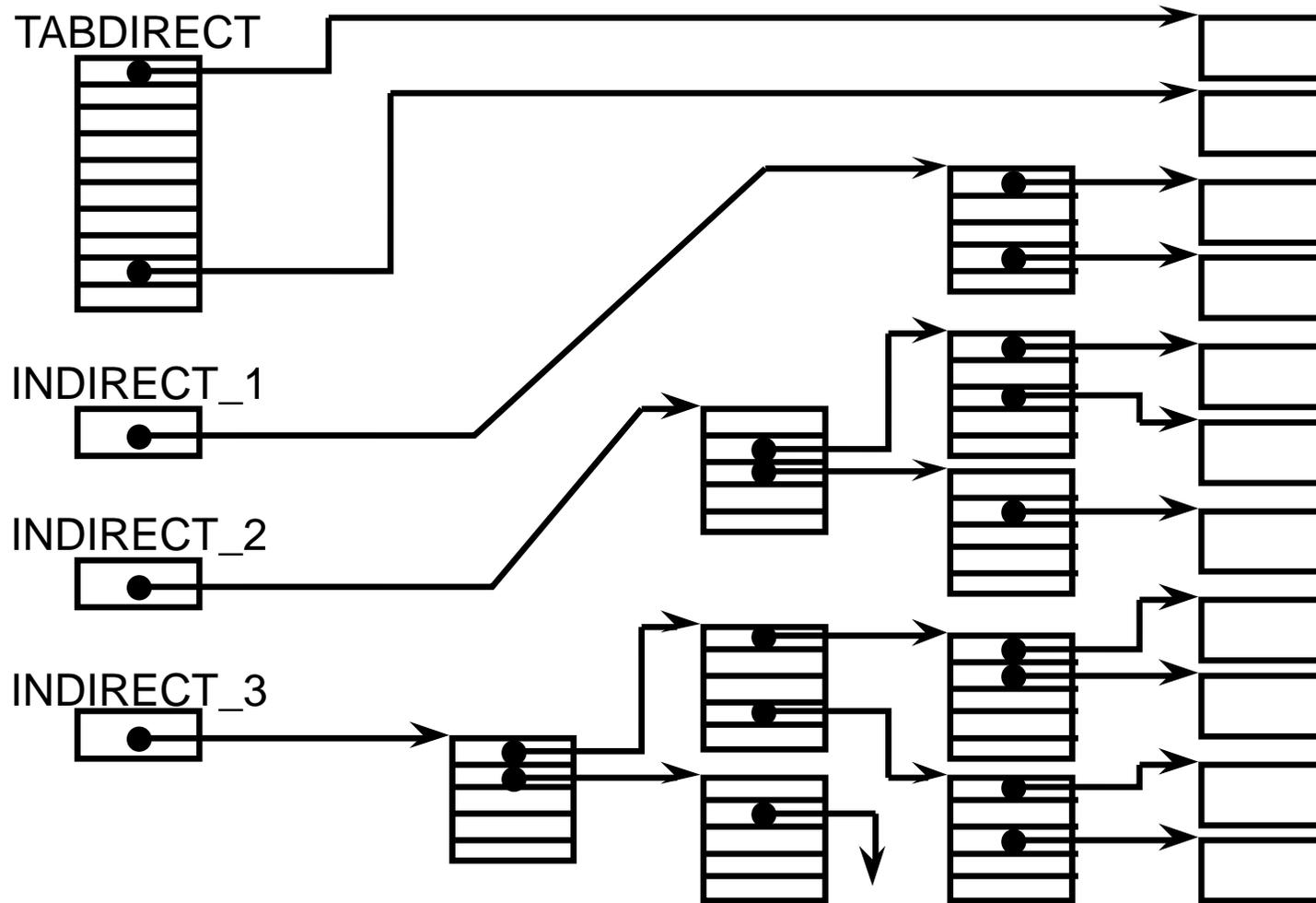
taille variable de cette table  $\Leftrightarrow$  pas de pénalisation pour les petits objets

UNIX:

- 10 premiers blocs de l'objet 10
- 1 numéro de bloc éventuel  
contenant les numéros de  $p$  blocs de données  $p$
- 1 numéro de bloc éventuel  
contenant les numéros de  $p$  blocs éventuels  
contenant les numéros de  $p$  blocs de données  $p^2$
- 1 numéro de bloc éventuel  
contenant les numéros de  $p$  blocs éventuels  
contenant les numéros de  $p$  blocs éventuels  
contenant les numéros de  $p$  blocs de données  $p^3$

taille maximum:  $10 + p + p^2 + p^3 = 16 \text{ Go}$  ( $p = 256$ )

# Allocation par bloc de taille fixe (4)



# Allocation par bloc de taille fixe (5)

```
r := i div nbsb;
si r < 10 alors j := TABDIRECT[r];
sinon r := r - 10;
    si r < p alors lire_bloc(INDIRECT_1, TAB_LOCALE);
        j := TAB_LOCALE[r];
    sinon r := r - p;
        si r < p * p alors
            lire_bloc(INDIRECT_2, TAB_LOCALE);
            lire_bloc(TAB_LOCALE[r/p], TAB_LOCALE);
            j := TAB_LOCALE[r mod p];
        sinon
            r := r - p * p;
            lire_bloc(INDIRECT_3, TAB_LOCALE);
            lire_bloc(TAB_LOCALE[r/(p*p)], TAB_LOCALE);
            r := r mod (p*p);
            lire_bloc(TAB_LOCALE[r/p], TAB_LOCALE);
            j := TAB_LOCALE[r mod p];
        finsi;
    finsi;
finsi;
nv := j * nbsb + i mod nbsb; → <ns, nf, nc>
```

# Représentation de l'espace libre (1)

- Quantum d'allocation: plus petite unité d'espace allouable  
secteurs consécutifs = blocs = clusters
- Influence de la taille du quantum  
perte d'espace: besoin N octets et alloué P quanta => dernier reçu utilisé à 50%  
nombre d'unités: disque M octets =>  $M/q$  unités allouables
- Systèmes à allocation par zone => besoins en quanta  
unité de base fixée par le système et connue des utilisateurs
  - nombre variable d'unités suivant le disquenombre maximum d'unités par disque, et besoins en quanta de base
  - quantum de taille variable suivant le disque, multiple du quantum de base
  - 256 Mo = 64000 quanta de 4 Ko; 5 Go = 78000 quanta de 64 Ko
- Système à allocation par bloc: quantum = taille du bloc

# Représentation de l'espace libre (2)

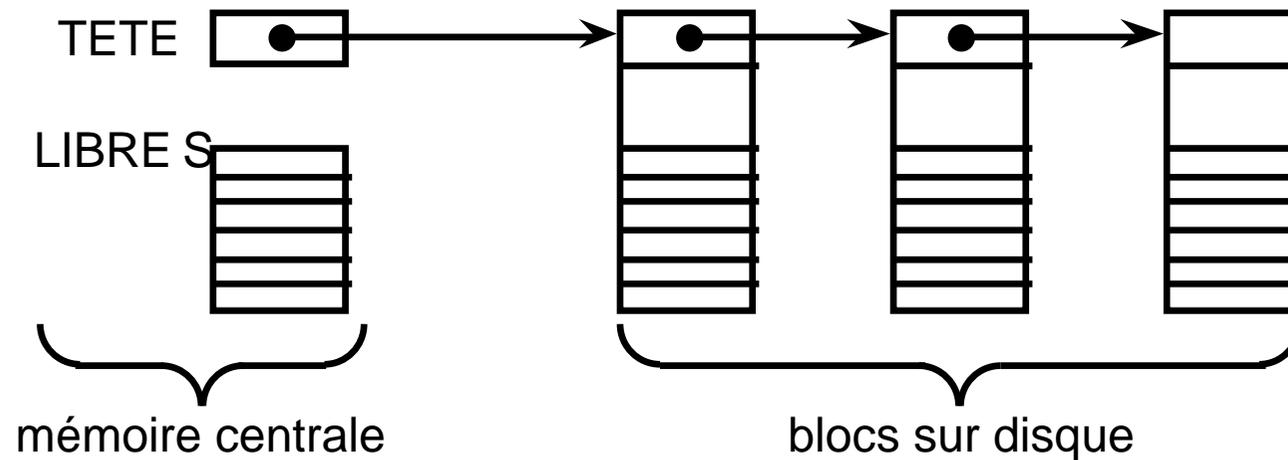
- Représentation par table de bits (DEC-VMS, NTFS,...)

```
tableau état[0..M] de (LIBRE, OCCUPÉ)
i := 0; -- mais on peut commencer n'importe où
trouvé := faux;
k := 0;
tant que i < N et non trouvé faire
    si état[i] = LIBRE alors
        si k = 0 alors j := i; finsi; -- repère début de zone
        k := k + 1;
        trouvé := k = p;           -- zone de taille suffisante
    sinon k := 0;                  -- zone trop courte
    finsi;
    i := i + 1;
fait;
si trouvé alors allouer de j à j + p - 1 sinon impossible finsi
```

*amélioration: liste partielle de zones libres*

# Représentation de l'espace libre (3)

- Par liste de zones libres (certains Unix)
- Taille variable => liste <adresse, longueur>
- Idée: utiliser l'espace libre pour y mettre sa représentation



# Conclusions

- Espace disque => linéarisation (numérotation virtuelle)
- Objet externe => numéro logique des secteurs (local à l'objet)  
représentation de l'espace: comment passer du numéro logique au numéro virtuel
- Allocation par zone => secteurs contigus, pas toujours possible  
implantation séquentielle, avec ou sans extensions
- Allocation par blocs => tous blocs de même taille  
implantation par chaînage, table centralisée  
implantation par table propre à l'objet, éventuellement à plusieurs niveaux
- Quantum d'allocation => mesure d'espace allouable  
fixe pour l'utilisateur, mais souvent dépendant du support
- Espace libre => table de bits ou liste de zones libres