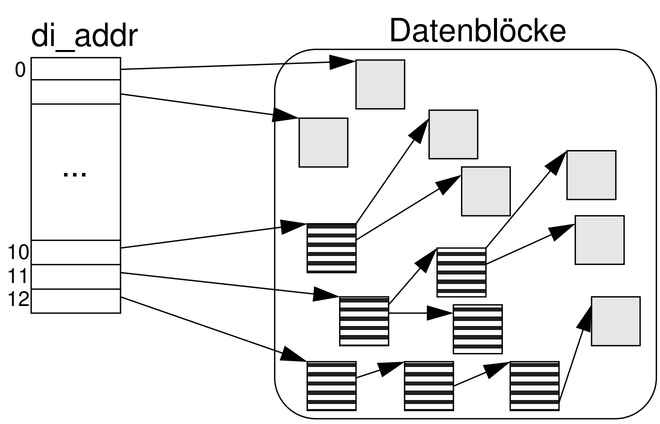
1 Liste der freien Datenblöcke

- allocate_datablock
  - suche nach Eintrag!=0 in erstem Block der Freeliste
  - wenn vorhanden, setze auf 0 und gib Eintrag zurück
  - wenn alle Einträge==0, klinke Block aus Liste aus und gib den Free-listen Block als Datenblock zurück
- reclaim_datablock(blocknr)
  - wenn es keinen ersten Freelisten-Block gibt, nimm Block blocknr dafür
  - sonst suche nach Eintrag==0 in erstem Block der Freeliste
  - wenn vorhanden, setze Eintrag auf blocknr
  - sonst verwende blocknr als neuen (ersten) Freelisten-Block

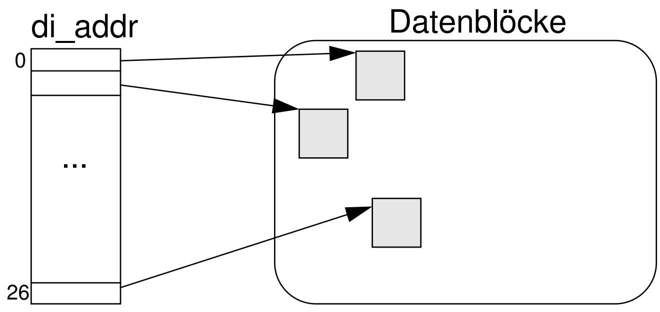
2 Die Blockliste einer Datei (di_addr)

- s5fs
  - ◆ Liste aus 13 physikalischen Blocknummern zu je 24 Bit
  - ◆ damit sind 2^{24} Blöcke = 8192 MByte = 8 GByte adressierbar
  - ◆ der 10. Eintrag wird verwendet für einfach indirekte Adressierung, der 11. für 2-fach indirekt, der 12. für 3-fach indirekt

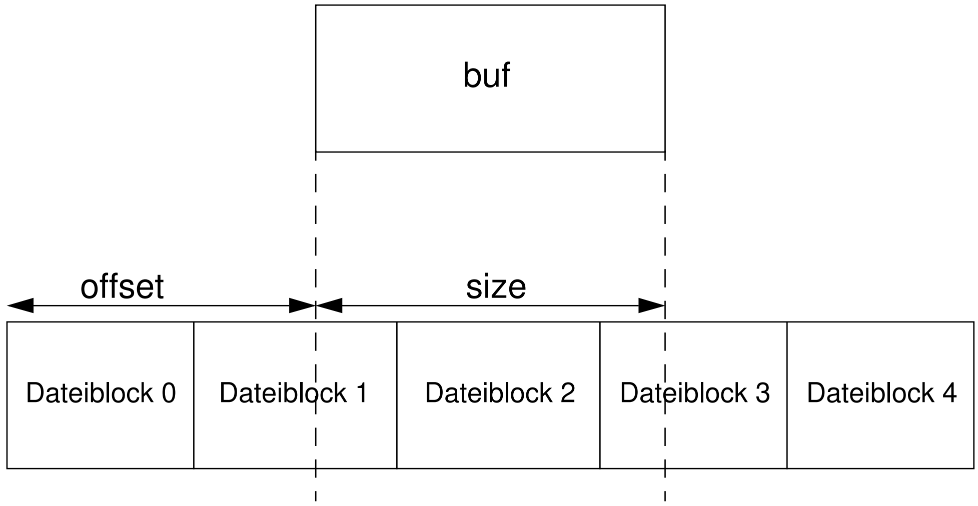


2.1 Die Blockliste einer Datei (di_addr)

- spfs
  - Liste aus 27 physikalischen Blocknummern zu je 32 Bit (= 4 Byte)
  - damit sind 2^{32} Blöcke = 2097152 MByte = 2048 GByte = 2 TByte adressierbar
  - in *spfs* nur direkte Adressierung, d.h. Maximallänge einer Datei ist 27* 512 Byte = 13,5 kByte



3 Dateien (sp_iread/sp_iwrite)



- bei sp_iwrite zu beachten:
  - keine Lücken in di_addr entstehen lassen
  - (Lücke: Dateiblock_n=0 und Dateiblock_{n+1}!=0)

3.1 sp_iread

- sp_iread(inodeno,buf,offset,size)
  - Inode einlesen
  - Blocknummer und Blockoffset entsprechend *offset* berechnen
  - *size* entsprechend Dateigröße einschränken
  - Anzahl der zu lesenden Daten = *size*
  - solange es noch Daten zu lesen gibt
    - physikal. Blocknummer aus di_addr holen (==0?)
    - Block lesen
    - Größe und Offset des zu kopierenden Block-Bereichs berechnen
    - Bereich aus Block nach *buf* kopieren, *buf* erhöhen
    - Anzahl der noch zu lesenden Daten um Größe des kopierten Bereiches erniedrigen
    - Blocknummer erhöhen
  - di_atime neu setzen und Inode auf Disk schreiben