Max size of a Small file is

\[ 8 \times 512 = 4096 \text{ Bytes} \]

To access byte \( N \) of a Small file:

\[ N = 0, \ldots, 4095 \]

\[ \left\lfloor \frac{N}{512} \right\rfloor = \text{Logical Block #3} \]

Fetch Contents of block addr[\( \left\lfloor \frac{N}{512} \right\rfloor \)]

\[ N \mod 512 = \text{offset} \]
Large file:

Block #775

First 512 bytes of the file

Block #811

Single Indirect Block (has 256 Block addresses)

Single Indirect Block

Data Block

Double Indirect Block

Max file size = \( 7 \times 256 \times 512 + 1 \times 256 \times 256 \times 512 \)

Size is stored in 24 bits.

\( 2^{24} = \text{max size} \)

\( 2^{24} = 16 \text{ MB} \)
Directory

Creates a mapping between names & i-nodes.

```
/venky
```

Fetch inode #11, it will be a directory [file type]
Fetch contents of this file [using addrs]

<table>
<thead>
<tr>
<th>i-node #</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Behappy</td>
</tr>
<tr>
<td>23</td>
<td>a.out</td>
</tr>
<tr>
<td>40</td>
<td>Behappy.h</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

2 bytes, 14 bytes, 16 bytes per entry

i-node # Name of file in that directory
Memory management

Looked at segmentation.

Suffers from fragmentation

[less than Dynamic pat-calls]

What if each "segment" of my program is of same size?

↓

Paging.

Partition each program into equal sized pages.

Page size is same for all programs

Partition RAM (physical memory)

into equal size frames:

frame size = page size = P

\[ P = 2^k \quad \text{k an int} \]
my program

Page Map Table

RAM

k bits

offset

Dynamic Address Translator

absolute address

1. Use page # and offset into page map table
2. Replace page # by f

my page number [address generated by my program]
Page Map Table

one / process

Store in PCB

when a process switch, switch
Page Map Table also.

A where to store - Page Map Table
while executing a program.

(i) Registers
Fast, expensive

(ii) RAM [ PCB]
Slow [100% ↑ in # of memory
access]

Not expensive

(iii) Hybrid scheme [Translation Look
aside buffer, Set Associative
storage ...]

Store few entries of the Page
Map Table in Registers.

<table>
<thead>
<tr>
<th>P_i</th>
<th>f_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_1</td>
<td>f_1</td>
</tr>
<tr>
<td>p_2</td>
<td>f_2</td>
</tr>
<tr>
<td>p_3</td>
<td>f_3</td>
</tr>
<tr>
<td>p_4</td>
<td>f_4</td>
</tr>
</tbody>
</table>

Say 4 entries of the Page Map Table in Registers.

| P_5 | 0 |

E.O. to translate
Search for P in parallel in hardware in TLB [Hybrid]

Success? [we have a hit]
Translation is fast

else
Fetch needed pMT entry
from RAM [PCB]
Replace one of the entries
of TLB by this entry

Translate

Choice of page size

words \( y = \text{Page Size, to be determined} \)

words \( S = \text{average program size} \)

words \( x = \# \text{words/entry of Page Map Table} \)

Memory overhead: \( \frac{y}{2} + \frac{S}{y} \cdot x \)
overhead

\[ w = \frac{y}{2} + \frac{sx}{y} \]

\[ \frac{dw}{dy} = \frac{1}{2} - \frac{sx}{y^2} = 0 \]

\[ y = \pm \sqrt{2sx} \]

\[ \frac{d^2w}{dy^2} = \frac{2sx}{y^3} \quad \text{is positive} \]

\[ y = \sqrt{2sx} \quad \text{minimizes} \quad w \]

Demand Paging

Program size > Max RAM available?

Only a part of the program needs to be in RAM to execute.
Store Complete program in say hard drive, indexed by Page #.

Part of the program is loaded in RAM

File Map Table    Page Map Table

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Disk address of page 3 of program

<table>
<thead>
<tr>
<th></th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>effective address</td>
</tr>
<tr>
<td></td>
<td>page (logical address)</td>
</tr>
</tbody>
</table>

Use p as index into pmT, find the entry needed: check if valid bit
if valid = 1 { Translate as before}
else { // Page fault
Consult File Map Table, initiate page p, can be brought into RAM

Switch to another process [this process is blocked]

<table>
<thead>
<tr>
<th>File Map Table</th>
<th>Page Map Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>only when E a page fault</td>
<td>Has one extra bit</td>
</tr>
<tr>
<td></td>
<td>(Valid bit) per entry (Page)</td>
</tr>
<tr>
<td>in PCB</td>
<td>PCB &amp; Some entries in Registers</td>
</tr>
</tbody>
</table>

Three policies:

A. Allocation Policy
   (i) Static
   (ii) Dynamic

B. Fetch Policy
   (i) Prefetch
   (ii) Demand Fetch

C. Replacement Policy