ECF

user process

exception

time

Control passed to exception handler

exception handler

Control returns to user process
1. Fault
   e.g. Page fault

   *Valid*

   ![Diagram with D, VPN, and PPN]

   Page fault exception handler.

   "Synchronous"

   ⇒ user process generated the fault.
2. Interrupts

"asynchronous"

user process

\[ \text{read()} \quad \text{system call} \]

user input
from keyboard
"interrupt"

waits for user input

continue user process
<table>
<thead>
<tr>
<th>Exception #</th>
<th>Desc.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>divide by 0</td>
<td>Fault</td>
</tr>
<tr>
<td>13</td>
<td>Protection fault</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>(segfault)</td>
<td>&quot;</td>
</tr>
<tr>
<td>14</td>
<td>Page fault</td>
<td>&quot;</td>
</tr>
<tr>
<td>128 (0x80)</td>
<td>System Call</td>
<td>Trap.</td>
</tr>
</tbody>
</table>
Exception table

EQ page fault
   exception #14.
**System Calls**

1. `exit()`  
2. `read()`  
3. `write()`  
4. `brk()`, `sbrk()`  
5. `open()`  
6. `close()`
<table>
<thead>
<tr>
<th>Syscall #</th>
<th>Name</th>
<th>generate/ create a new process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>exit</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>fork</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>write</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>open</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>close</td>
<td></td>
</tr>
</tbody>
</table>
int main()
{
    write(1, "hello", 6);
    exit(0);
}

fd for stdout
1. Syscall \# \rightarrow ./eax.

2. Store the args of the syscall in registers:
   - ./ebx
   - ./ecx
   - ./edx
   - ./esi
   - ./edi
   - ./ebp
Main memory