Process Creation / Termination

Termination

- Process issues an exit instruction

Cooperating processes need to communicate

Why?

- Sharing data
- Computational Speedup
- Modularity

Information Sharing two+ processes share the same piece of info

Computational Speedup

Modularity

Examples

- Producer / Consumer Model

```
co
// - Parrelization
oc
```

When referencing concurrency, we use the term 'arms' to describe the number of processes. A program can have two or MORE arms.

Independent: 'Can operate without each other, and complete its task w/o intervention of something else getting in the way'

```
string line;
read a line from stdin into line;
while( !EOF) {
    co look for pattern in line;
        if (pattern is in line)
            write lin;
    // read next line of input into line
    oc;
}
```

The above example is <u>not</u> independent, as they both use the line buffer.

```
string line1, line2;
read a line from stdin into line1;
while( !EOF) {
    co look for pattern in line1;
        if (pattern is in line1)
            write line1;
    // read next line of input into line2
    oc;
    line1 = line2;
}
```

What did we change?

- At the end of each loop iteration, copy the contents of `line2` into `line1`.

Is this solution efficient?

At each iteration of the while loop, how many concurrent processes are created, completed, and destroyed? 2

How do you determine if two processes are independent?

Read set of variables read by a process

Write set of variables written by a process

Separate processes P_a, P_b on separate CPUs

A process/thread may have a private set of variables scoped only to that thread.

Two processes are **independent** if the write set of each is disjoint from both the read and write sets of the other.

- Independence
- Threads
- Synchronization