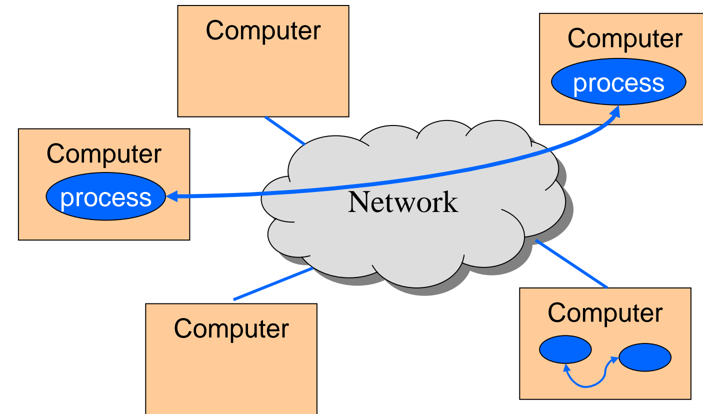
Inter-process Communication

CS 217

Networks

- Mechanism by which two processes exchange information and coordinate activities

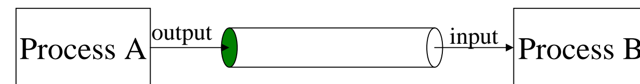


Inter-process Communication

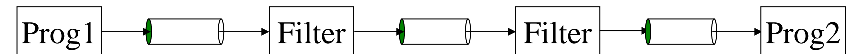
- Pipes
  - Processes must be on same machine
  - One process spawns the other
  - Used mostly for filters
- Sockets
  - Processes can be on any machine
  - Processes can be created independently
  - Used for clients/servers, distributed systems, etc.

Pipes

- Provides an interprocess communication channel



- A filter is a process that reads from **`stdin`** and writes to **`stdout`**



Pipes (cont)

- Many Unix tools are written as filters
  - **grep, sort, sed, cat, wc, awk ...**
- Shells support pipes

```
ls | wc -l
who | grep mary | wc -l
cat < foo | grep bar | sort > save
```

- The combination of these features gives Unix incredible power and flexibility:
  - **Standard I/O**
  - **I/O redirection**
  - **Pipes**

The genesis of pipes

10

Summary--what's most important.

To put my strongest concerns in a nutshell:

1. We should have some ways of coupling programs like garden hose--screw in another segment when it becomes then it becomes necessary to massage data in another way. This is the way of IO also.

2. Our loader should be able to do link-loading and controlled establishment.

3. Our library filing scheme should allow for rather general indexing, responsibility, generations, data path switching.

4. It should be possible to get private system components (all routines are sytem components) for buggering around with.

M. D. McIlroy
Oct. 11, 1964

Pipes and pipelines

- **connect output of one program to input of another**
  - who | grep joe | wc
- **Ken's idea, with Doug's prodding?**
  - Ken's notation?
- **frenzy of invention**
  - reformulating programs to work in pipelines
  - e.g., sort, and why it can't be in a pipeline
- **spell program**
  - cat *files* | tr ... | sort | uniq | comm -1 - dict
- **modularization of programs**
  - grap | pic | tbl | eqn | troff

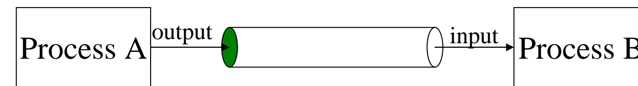
grep: the quintessential tool

- **"Reach out and grep someone"**
- **genesis from ed editor**
- **programmer's tool**
- **building block in scripts**
- **dictionary searches**

```
grep '^[behilos]*$' /usr/dict/web2
```

```
b       bilsh   bose    hellish ibisbil liss    ohelo   shies   slish
be      bios    bosh    hello   ie      lo      oho     shih    slob
bebless biose   boss    heloe   ihi     lob     oii     shill   sloe
beboss  biosis  e       helosis ill     lobbish oil     shilloo sloo
bee     bis     ebb     hi      illess  lobe    oilhole shish   sloosh
beeish  bleb    eboe    hie     illish  lobeles oilless sho     slosh
beelol  blee    eel     hill    io      lobo    oleo    shoe    so
bees    bleo    eelbob  his     is      lobose  oleose  shoebil sob
bel     bless   eh      hish    isle    loess   olio    shoeles soboles
belee   blibe   el      hiss    isleles loll    os      shole   soe
belibel bliss   elb     ho      iso     loo     ose     shoo    soh
belie   blissle ell     hob     isohel  loose   osse    shooi   soho
bell    blo     elle    hobbil  issei   loosish s       shool   soil
belle   blob    els     hobble  l       lose    se      si      soilles
bes     bo      else    hobo    lee     losel   see     sib     sol
besee   bob     es      hoe     lees    losh    seel    sie     sole
beshell bobbish ess     hoi     lei     loss    seesee  sil     soleil
besoil  bobble  h       hoise   less    lossles seise   sile    soleles
bib     bobo    he      hole    lessee  o       sele    sill    soles
bibb    boho    heel    holeles li      obe     sell    silo    soli
bibble  boil    heelles holl    libel   obese   sellie  siol    solio
bibi    bole    hei     hollo   libelee obi     sess    sis     solo
bibless bolis   heii    hoose   lie     oboe    sessile sise    sool
bilbie  boll    helbeh  hoosh   liesh   obol    sh      sisel   sooloos
bilbo   bolo    hele    hose    lile    obole   she     sish    sosh
bile    boo     helio   hosel   lill    obsess  shee    sisi    soso
bilio   boob    heliosi hoseles lis     oe      shell   siss    sosoish
bill    boohoo  hell    i       lish    oes     shi     sissoo  soss
bilo    bool    hellhol ibis    lisle   oh      shiel   slee    sossle
bilobe  boose
```


Creating a Pipe



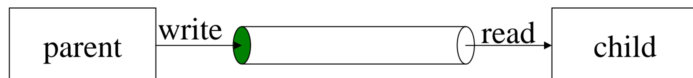
- Pipe is a communication channel abstraction
  - Process A can write to one end using "write" system call
  - Process B can read from the other end using "read" system call
- System call

```
int pipe( int fd[2] );
return 0 upon success –1 upon failure
fd[0] is open for reading
fd[1] is open for writing
```

- Two coordinated processes created by **fork** can pass data to each other using a pipe.

Pipe Example

```
int pid, p[2];
...
if (pipe(p) == -1)
   exit(1);
pid = fork();
if (pid == 0) {
   close(p[1]);
   ... read using p[0] as fd until EOF ...
}
else {
    close(p[0]);
    ... write using p[1] as fd ...
    close(p[1]); /* sends EOF to reader */
    wait(&status);
}
```



Dup

- Duplicate a file descriptor (system call)

```
int dup( int fd );
```

  duplicates **fd** as the lowest unallocated descriptor

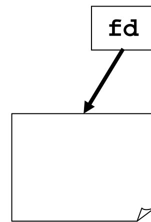
- Commonly used to implement redirection of stdin/stdout
- Example: redirect stdin to "foo"

```
int fd;
fd = open("foo", O_RDONLY, 0);
close(0);
dup(fd);
close(fd);
```

Dup

- Duplicate a file descriptor (system call)
  `int dup( int fd );`
  duplicates **fd** as the lowest unallocated descriptor
- Commonly used to implement redirection of stdin/stdout
- Example: redirect stdin to “foo”

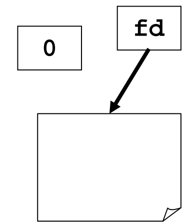
```
int fd;
fd = open(“foo”, O_RDONLY, 0);
close(0);
dup(fd);
close(fd);
```



Dup

- Duplicate a file descriptor (system call)
  `int dup( int fd );`
  duplicates **fd** as the lowest unallocated descriptor
- Commonly used to implement redirection of stdin/stdout
- Example: redirect stdin to “foo”

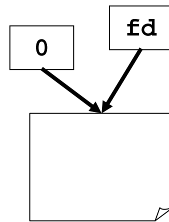
```
int fd;
fd = open(“foo”, O_RDONLY, 0);
close(0);
dup(fd);
close(fd);
```



Dup

- Duplicate a file descriptor (system call)
  `int dup( int fd );`
  duplicates **fd** as the lowest unallocated descriptor
- Commonly used to implement redirection of stdin/stdout
- Example: redirect stdin to “foo”

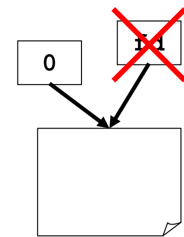
```
int fd;
fd = open(“foo”, O_RDONLY, 0);
close(0);
dup(fd);
close(fd);
```



Dup

- Duplicate a file descriptor (system call)
  `int dup( int fd );`
  duplicates **fd** as the lowest unallocated descriptor
- Commonly used to implement redirection of stdin/stdout
- Example: redirect stdin to “foo”

```
int fd;
fd = open(“foo”, O_RDONLY, 0);
close(0);
dup(fd);
close(fd);
```



Dup2

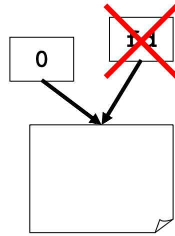
- For convenience…

```
dup2( int fd1, int fd2 );
```

use `fd2(new)` to duplicate `fd1 (old)`
closes `fd2` if it was in use

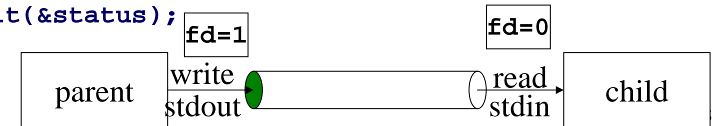
- Example: redirect stdin to "foo"

```
fd = open("foo", O_RDONLY, 0);
dup2(fd,0);
close(fd);
```



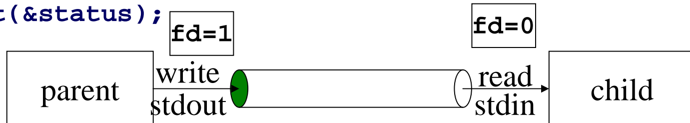
Pipes and Standard I/O

```
int pid, p[2];
if (pipe(p) == -1)
   exit(1);
pid = fork();
if (pid == 0) {
   close(p[1]);
   dup2(p[0],0);
   close(p[0]);
   ... read from stdin ...
}
else {
   close(p[0]);
   dup2(p[1],1);
   close(p[1]);
   ... write to stdout ...
   wait(&status);
}
```



Pipes and Exec()

```
int pid, p[2];
if (pipe(p) == -1)
   exit(1);
pid = fork();
if (pid == 0) {
   close(p[1]);
   dup2(p[0],0);
   close(p[0]);
   execl(...);
}
else {
   close(p[0]);
   dup2(p[1],1);
   close(p[1]);
   ... write to stdout ...
   wait(&status);
}
```



Unix shell (sh, csh, bash, ...)

- Loop
  - Read command line from stdin
  - Expand wildcards
  - Interpret redirections < > |
  - pipe (as necessary), fork, dup, exec, wait
- Start from code on previous slides, edit it until it's a Unix shell!