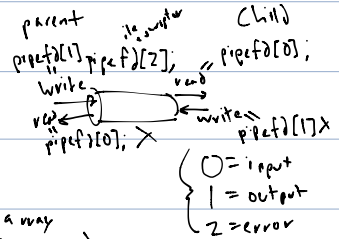
```
pid = fork()
if( pid == 0) {child}
    pid = fork(); / execvp(...);
}
else { parent
 pid=fork();
  if (pid==0) {child
```



pass info from one process to another

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

pipe(pipefd);
```

cat scores | grep uva

```
int main(int argc, char** argv)
{
    int pipefd[2];
    int pid;
    char* cat_args[] = { "cat", "scores", NULL }
    char* grep_args[] = { "grep", "uva", NULL};
     pipe(pipefd); // if( pipe(pipefd) != 0)
     pid=fork();
     if(pid==0)
     { dup2( pipefd[0], 0);
       close(pipefd[1]);
       execvp("grep", grep_args);
     }
     else
     { dup2( pipefd[1], 1);
```

```
        close(pipefd[0]);
        execvp("cat", cat_args);
    }
}
```

I/O redirection

same header files

inputfile outputfile
grep uva <scores> out

```
int infile, outFile;
char *grep_args[] = {"grep", "uva", NULL};
{ k?
inFile = open("scores", O_RDONLY);
outFile = open("out", O_WRONLY);
dup2(inFile, 0);   // scores
dup2(outFile, 1);  // out
close(inFile);
close(outFile);
execvp("grep", grep_args);
}
```

c program wordcount
cat <redirect.c | wc > out

```
int main(int argc, char **argv)
{
    int inFile, outFile;
```

```
char *cat_args[] = {"cat", NULL};
char *wc_args[] = {"wc", NULL};
int status, i;
pid_t first, second;
  int pipefd[2];
inFile = open("redirect.c", O_RDONLY);
outFile = open("out", O_WRONLY);
pipe(pipefd);
 first = fork();
 if (first == 0)
 { dup2(inFile, 0);   // redirect.c
    dup2(pipefd[1], 1);
    close(pipefd[0])
     execvp("cat", cat_args);      <- didnt include check errors need for project
  }
  else if (first > 0)
  { second = fork();
     if (second == 0)
     { dup2(pipefd[0], 0);
        dup2(outFile, 1);   // =out
        close(pipefd[1]);
        execvp("wc", wc_args);
      }
    else if (second > 0)
    { close(inFile);
```

```
    closeOutFile();
    close(pipefd[0]);
    close(pipefd[1]);
    for (i=0; i<2; i++)
        wait(&status)
    print("All done\n");
    }
  }
}
```

process control block PCB

(1) state transition 1 : 1 cycle for process

active or suspended

ready, running, block

"waiting for event to occur" signal

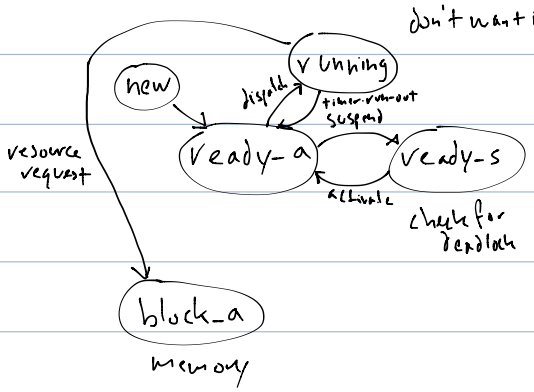
CPU bound

I/O bound

process

don't want idle CPU or I/O

multilevel queue system



CPU Schedule "schedules"

dispatcher "context switch"

quantum = time slice of time

1s, 1/10s