fork: Creating new processes

int fork(void)

creates a new process (child process) that is identical to the calling process (parent process)

* returns 0 to the child process
* returns child’s pid to the parent process

if (fork () == 0)

{

printf ("hello from child\n");

}

else

{

printf ("hello from parent\n");

}

Fork Example #1

Key Points

* Parent and child both run same code
* Distinguish parent from child by return value from fork
* Start with same state, but each has private copy

void fork1()

{

int x = 1;

pid\_t pid = fork();

if (pid == 0)

{printf ("Child has x = %d\n", ++x); }

else {printf("Parent has x = %d\n", --x); }

printf ("Bye from process %d with x = %d\n", getpid(), x); }

Fork Example #2

Both parent and child can continue forking

void fork2()

{

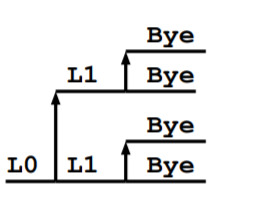
printf("L0\n");

fork();

printf("L1\n");

fork();

printf("Bye\n"); }



Fork Example #3

void fork4()

{ printf("L0\n");

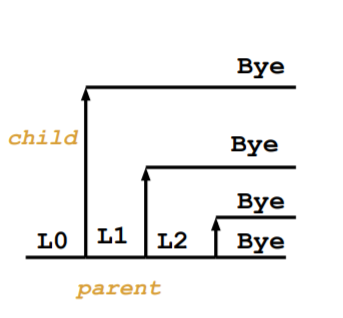
if (fork() != 0)

{ printf("L1\n");

if (fork() != 0)

{ printf("L2\n"); fork(); } }

printf("Bye\n"); }



exit: Destroying Process

void exit(int status)

* exits a process
* Normally return with status 0

atexit() registers functions to be executed upon exit

void cleanup(void)

{ printf("cleaning up\n"); }

void fork6() {

atexit(cleanup);

fork();

exit(0); }