

#0 Android source code

https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/host/i686-linux-glibc2.7-4.6/+tools_r20/sysroot/usr/include/bits/waitstatus.h

```
/* If WIFEXITED(status), the low-order 8 bits of the status. */
#define __WEXITSTATUS(status) (((status) & 0xff00) >> 8)
```

```
/* If WIFSIGNALED(status), the terminating signal. */
#define __WTERMSIG(status) ((status) & 0x7f)
```

```
/* If WIFSTOPPED(status), the signal that stopped the child. */
#define __WSTOPSIG(status) __WEXITSTATUS(status)
```

```
/* Nonzero if STATUS indicates normal termination. */
#define __WIFEXITED(status) (__WTERMSIG(status) == 0)
```

#1 Review - Can we ensure SECRET is saved to the log file?

```
01 close(1); // goodbye standard out
02 open("log.txt", O_RDWR | O_CREAT | O_APPEND, 0644);
03 puts("SECRET");
04 ?
05 execlp("/bin/ls", "ls", getEnv("HOME"), (char*)NULL);
```

#2 The fork-exec-wait trilogy

fork. Are variables shared?

exec. When does exec return?

waitpid. Waiting for your child?

#3 What happened to your child? - use the wait macros to extract bits

```
pid_t waitpid(pid_t pid, int * status, int options);
```

//Decoding the bits of the status integer

```
01 int s;
02 waitpid(child, &s, 0);
03 WEXITSTATUS(s) valid if WIFEXITED(s) != 0
04 WTERMSIG(s) valid if WIFSIGNALED(s) != 0
```

#4 Who is my parent?

```
01 pid_t vader = getppid();
02 pid_t luke = getpid();
```

#5 Review - How does sleeps sort work?

```
01 int main(int c, char **v) {
02     while (--c > 1 && !fork());
03     int val = atoi(v[c]);
04     sleep(val);
05     printf("%d\n", val);
06     return 0;
07 }
```

#6 Puzzle - Two processes for the price of one program

```
01 char * m = "World";
02 int main() {
03     int a = 0;
04     pid_t f = fork();
05     if(f == -1) { perror("fork failed!"); exit(1);}

06     if(_____) { /* child process */ m = "Hello";}

07     else { // I'm the parent
08         printf("Waiting for %ld to finish", (long)f);

09     ?

10     ?

11     }
12     puts(m);
13     return 42;
14 }
```

Post lecture challenge 1. Write a forking program where the parent process creates N child processes.

or...

Post lecture Challenge 2. Write a forking program that creates a chain of N processes i.e. each process, except the last, has one child process. (See if you can work this out yourself first before looking at my svn example)

#7 A program to automatically compile and execute my programs

```
01 char * compiler = "gcc";
02 int main(int argc, char** argv) {
03     if(argc != 2) {
04         fprintf(stderr, "%s prog.c", argv[0]);
05         exit(1);
06     }
07     char* target = argv[1];
08     while(1) {
09         pid_t child = fork();
10
11         if(_____) { // I'm the child
12             execlp(
13                 perror(compiler);
14                 exit(1);
15             }
16             int status=0;
17
18             if(_____) break;
19             sleep(5);
20         }
21         puts("running your program"); // no flush!?
22         execlp("./a.out", "./a.out", (const char*)NULL);
23         perror("Failed to run ./a.out");
24         return 1;
25     }
```

#8 What happens to child processes if their parents die first?

#9 What happens if the parent never finishes and never waits on its children?

#10 What is SIGCHILD ?

#11 C Review / FAQ

What is special about sizeof(char) ?

int *x = & 0x12340;

On a 32 bit machine, what is the value of (x + 1) ?

Spot the mistake(s)!

```
01 double *a = malloc( sizeof(double*) );
02 double *b = a;
03 free(b); b = 0;
04 *a = (double) 0xbaadf00d;
05
06 char* result;
07 strcpy(result, "CrashMaybe");
08
09 void* append(char** ptr, const char*mesg) {
10     if(!*ptr) ptr = malloc( strlen(mesg) );
11     strcat( *ptr, mesg);
12 }
```