|  |  |
| --- | --- |
| CS 341 | Lecture Handout #6 |

#1 The fork and wait pattern

<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)

#2 The fork-exec-wait trilogy

fork Are variables shared?

exec When does exec return?

waitpid Purpose?

#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   1. int s; 2. waitpid(child, &s, 0 ); 3. WEXITSTATUS(s) valid if WIFEXITED(s) != 0 4. WTERMSIG(s) valid if WIFSIGNALED(s) != 0 |

#4 Who is my parent?

|  |
| --- |
| 1. pid\_t vader = getppid(); 2. pid\_t luke = getpid(); |

#5 Madness- What does this do and how?

|  |
| --- |
| 1. int main(int c, char \*\*v) { 2. while (--c > 1 && !fork()); 3. int val = atoi(v[c]); 4. sleep(val); 5. printf("%d\n", val); 6. return 0; 7. } |

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

|  |
| --- |
| 1. char \* m = "World"; 2. int main() { 3. int a = 0; 4. pid\_t f = fork(); 5. if(f == -1) { perror("fork failed!"); exit(1);} 6. if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {/\* child process \*/ m = "Hello";} 7. else { // I'm the parent 8. printf("Waiting for %ld to finish", (long)f); 9. ? 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

|  |
| --- |
| 1. char \* compiler = "gcc"; 2. int main(int argc, char\*\* argv) { 3. if(argc != 2) { 4. fprintf(stderr,"%s prog.c",argv[0]); 5. exit(1); 6. } 7. char\* target = argv[1]; 8. while(1) { 9. pid\_t child = fork(); 10. if(\_\_\_\_\_\_\_\_\_\_){ // I'm the child 11. execlp( 12. perror(compiler); 13. exit(1); 14. } 15. int status=0; 17. if( ) break; 18. sleep(5); 19. } 20. puts("running your program"); // no flush!? 21. execlp("./a.out","./a.out",(const char\*)NULL); 22. perror("Failed to run ./a.out"); 23. return 1; 24. } |

#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)!

|  |
| --- |
| 1. double \*a = malloc( sizeof(double\*) ); 2. double \*b = a; 3. free(b); b = 0; 4. \*a = (double) 0xbaadf00d; 5. char\* result; 6. strcpy(result, "CrashMaybe"); 7. void\* append(char\*\* ptr, const char\*mesg) { 8. if(!\*ptr) ptr = malloc( strlen(mesg) ); 9. strcat( \*ptr, mesg); 10. } |