

What does the following 'exec' example do?

```
int main() {
    close(1); // close standard out
    open("log.txt", O_RDWR | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR);
    puts("Captain's log");
    chdir("/usr/include");
    execl("/bin/ls", "/bin/ls", ".", (char*)NULL); // "ls ."
    perror("exec failed");
    return 0; // Not expected
}

int main(int argc, char**argv) {
    srand(time(NULL));
    pid_t child = fork();
    int r = rand() & 0xf;
    printf("%d: My random number is %d\n", getpid(), r);
}
```

What does the following program do and how does it work?

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

What does the child inherit from the parent?

What is different in the child process than the parent process?

How do I wait for my child to finish?

Can I find out the exit value of my child?

How do I start a background process?

*Remember ! Good parents don't let their children become zombies!*

What would be effect of too many zombies?

What does the system do to help prevent zombies?

How do I prevent zombies?

C Puzzle: Spot the error(s)!

```
char* f() {  
    char result[16];  
    strcat( result, "Hi");  
    int *a;  
    if( &a != NULL) { printf("Yes %d\n",42); }  
  
    struct link* first= malloc(sizeof(struct link*));  
    free(first)  
    if(first->next) free(first->next);  
    return result;  
}
```

Debugging tips I:

- malloc gotchas: Initialized? Sufficient
- sizeof gotchas:
- pointers assign before use?
- free gotchas
- beware of casting
- pointer arithmetic
- gdb backtrace

If there's time...

How do I send signals programmatically to a process?

How do I send a user-defined signal? Terminate signal?