
CS 4061: Practice Exam 1 SOLUTION

Spring 2021

University of Minnesota

Exam period: 30 minutes

Points available: 40

Problem 1 (10 pts): Examine the code to the right and describe what you expect its output to be. Explain why or why not you would expect to see any specific ordering in the output of the program.

SOLUTION: The processes will fork out in a "line" or "list" rather than branching in a tree. This is because each parent process falls into the if() consequence and will break out of the loop so will have only a single child. The order of output will start with the last child in the last iteration and proceed backwards to the original parent due to the placement of wait().

Example:

```
> a.out
```

```
iter 4, 13374 from 13373
iter 3, 13373 from 13372
iter 2, 13372 from 13371
iter 1, 13371 from 13370
iter 0, 13370 from 26018
```

Problem 2 (10 pts): Write the function `total_doubs()` described and demonstrated below.

```
1 #include <stdio.h>
2 int total_doubs(char *fname, // file to read
3                 double *total); // set to total
4 // Read doubles from the named file which contains
5 // binary doubles. Sum all the numbers in the file
6 // and set the double pointed to by total to this
7 // value. Return the count of numbers or -1 if
8 // the file could not be opened.
9
10 int main(){
11     char *fname = "nums.dat";
12     int n_doubs = -1;
13     double total = -1;
14     n_doubs = total_doubs("nums.dat", &total);
15     printf("%d nums read totaling %lf\n",
16           n_doubs, total);
17     return 0;
18 }
```

```
#include "headers.h" // standard headers
int main(){
    for(int i=0; i<5; i++){
        pid_t p = fork();
        if(p != 0){
            wait(NULL);
            printf("iter %d, %d from %d\n",
                i, getpid(), getppid());
            fflush(stdout);
            break;
        }
    }
    exit(0);
}
```

```
1 #include "total_doubs.h" SOLUTION
2 #define BUFCOUNT 4
3 #define BUFSIZE (sizeof(double) * BUFCOUNT)
4
5 int total_doubs(char *fname, double *total){
6     int fd = open(fname, O_RDONLY);
7     if( fd == -1 ){
8         perror("Couldn't open file");
9         return -1;
10    }
11    double buf[BUFCOUNT], tot = 0.0;
12    int total_bytes = 0;
13    while(1){
14        int bytes_read = read(fd, buf, BUFSIZE);
15        if(bytes_read == 0){
16            break;
17        }
18        total_bytes += bytes_read;
19        int count = bytes_read / sizeof(double);
20        for(int i=0; i<count; i++){
21            tot += buf[i];
22        }
23    }
24    close(fd);
25    *total = tot;
26    return (total_bytes / sizeof(double));
27 }
```

Problem 3 (5 pts): Stan Dardin is interactively testing his implementation of `commando` by punching in commands like `list`, `output-for 2`, `gcc test.c` by hand. He is finding that he keeps making mistakes while entering commands causing him to have to restart the program. He is not ready to run the full tests provided by Prof. Coffmalevolent, just the few things he knows works but the tedium is making it hard to proceed. Suggest an easy way for Stan to enter his commands more easily using tools and syntax provided in every Unix shell.

SOLUTION: Place the commands in a text file then use output redirection to feed the commands to `commando`. Either of these would work:

```
./commando --echo < input.txt OR cat input.txt | ./commando --echo
```

Background: If a `commando` user mistypes the name of a program like `fcc` rather than `gcc` strange output might occur. The next problems address this by making changes to `commando` so that it behaves like the example to the right.

Problem 4 (15 pts): Standard output for `commando` programs is directed into a pipe but standard error is not. Add code to `cmd_start()` in the template given which directs standard error into the output pipe associated with the `cmd_t`. Add code to the template for `cmd_update_state()` that checks for special exit codes that signify exec problems and sets fields of the `cmd_t` appropriately. Some code for this problem may be identical to what is already in your project.

```
> commando
@> fcc
@>
@!!! fcc[#9620]: EXEC FAIL
@> flarb
@>
@!!! flarb[#9621]: EXEC FAIL
@> list
JOB  #PID      STAT   STR_STAT  OUTB  COMMAND
0    #9620     128   EXEC FAIL   52   fcc
1    #9621     128   EXEC FAIL   52   flarb
@> output-for 0
@<<< Output for fcc[#9620] (52 bytes):
-----
Cmd failed to start exec: No such file or directory
-----
@> exit
>
```

```
----- SOLUTION -----
1 #define EXEC_FAIL 128
2 void cmd_start(cmd_t *cmd){
3     ... // VARIOUS SETUP
4     cmd->pid = fork();
5     ...
6     if(cmd->pid != 0){...} // PARENT
7     else{ // CHILD
8         // normal standard output redirection set up
9         ...
10        // YOUR CODE FOR STDERR REDIRECTION HERE
11        dup2(cmd->output_pipe[PWRITE], STDERR_FILENO);
12
13        // FINISHED WITH STDERR REDIRECTION
14
15        // Call to execvp(), on error, call perror()
16        // and exit with code EXEC_FAIL.
17        // YOUR CODE FOR EXEC/RETURN HERE
18
19        int ret = execvp(...);
20        if(ret == -1){
21            perror("Exec failed");
22            exit(EXEC_FAIL);
23        }
24    }
25 }
26 }
27 }
```

```
----- SOLUTION -----
1 #define EXEC_FAIL 128
2 void cmd_update_state(cmd_t *cmd, int block){
3     ... // VARIOUS SETUP
4     pid_t ret = waitpid(cmd->pid, &status, block);
5     ...
6     // CHECK IF status INDICATES CHILD HAS EXITED
7     // SET DATA ASSOCIATED WITH IT APPROPRIATELY
8     // CHECK FOR SPECIAL RETURN VALUE EXEC_FAIL
9     // AND SET THE str_status FIELD TO "EXEC FAIL"
10    // OTHERWISE SET DATA AS NORMAL IN PROJECT
11    if( ..check for exit as normal.. ){
12        int exit_status =
13            ..use macro to set status with exit code..;
14        cmd->status = exit_status;
15        cmd->finished = 1;
16
17        if(exit_status == EXEC_FAIL){
18            sprintf(cmd->str_status, STATUS_LEN, "EXEC FAIL");
19        }
20
21        else{
22            ..set str_status as normal..;
23        }
24
25    } // FINISHED CHECKING ON EXIT
26    ... // PRINT ALERTS
27 }
28 }
```

(Note: In the actual exam, more space will be provided for answers.)