

Homework 8

Problem 1. Read the following codes and answer the questions:

```
1  #include "csapp.h"
2
3  int main()
4  {
5      pid_t pid;
6      int x = 5;
7      int i;
8
9      FILE *f = fopen("a", "r+");
10
11     pid = Fork();
12
13     if (pid == 0) {
14         ++x;
15         fclose(f);
16         exit(0);
17     }
18
19     fprintf(f, "%d\n", --x);
20     fclose(f);
21     exit(0);
22 }
```

Questions:

1. By which process will the statement at line 14 be executed? (Parent/Child)
2. What will be printed to the file 'a'?
3. What happens if line 19 is executed **after** line 15?

Solutions:

1. Child.
2. 4
3. Nothing happens.

Problem 2. Read the following codes and answer the questions:

```
1  #include "csapp.h"
2
3  int main()
4  {
5      pid_t pid;
6
7      pid = Fork();
```

8	if (pid == 0) exit(0);
9	
10	Fork();
11	
12	pid = Fork();
13	if (pid == 0) exit(0);
14	
15	Fork();
16	
17	printf("hello\n");
18	
19	exit(0);
20	}

Questions:

1. How many times is Fork() called?
2. How many lines are printed?

Solutions:

1. $1+1+2+2=6$
2. 4

Problem 3. Fill in the blanks in the following codes. (Hint: you may refer to the slide '44-wait' and study the use of environment variable 'PATH' first)

1	<pre> #include "csapp.h" #define MAXLEN 1000 int main() { char *bin = "ls"; char *argv[] = { bin, NULL }; char buf[MAXLEN]; /* Fetch the environment variable 'PATH' into 'env_path' */ char *env_path = __1__; strcpy(buf, env_path); char *p = buf; char *delim; char path[MAXLEN]; /* Paths are separated by colon ':' in 'PATH'. */ while ((delim = __2__)) { *delim = '\0'; </pre>
---	---

	<pre> strcpy(path, p); /* Suppose now 'path' is '/usr/bin', * how do we obtain '/usr/bin/ls' in 'path'? * Hint: use 'strcat'. */ __3__ /* Execute the command in 'path' */ __4__ /* Make 'p' point to the next path string */ __5__ } fprintf(stderr, "No binary named %s found\n", bin); exit(1); } </pre>
--	---

Questions:

1. What does the program above do? Try it on your Linux!
2. Is it necessary to check the return value of 'execve'? Why?
3. What does it mean if 'execve' returns an error?

Solutions:

1. `getenv("PATH")`
2. `strchr(p, ':')`
3. `strcat(strcat(path, "/"), bin);`
4. `execve(path, argv, NULL);`
5. `p = delim + 1;`

1. Search for 'ls' in 'PATH' and execute it without arguments, which lists all visible files in the current directory.
2. No because if 'execve' succeeds, it will never return.
3. It means the program 'ls' is not in 'path'.

Problem 4. Write a program to create a child process. The child process should **wait** for signals to come. When the parent sends a SIGINT to its child, the child should print a line 'Hello' and exit.

Solution:

```

#include "csapp.h"

void handler(int sig)
{
    printf("Hello\n");
    exit(0);
}

```

```
int main()
{
    if (signal(SIGINT, handler) == SIGERR)
        unix_error("signal error");

    pid_t pid;

    if ((pid = Fork()) == 0)
        pause();

    Kill(pid, SIGINT);
    exit(0);
}
```