Sample solution for Quiz 1~3

Quiz#1

1.

Ans: 101 201 202

Note that the compiler optimizations don't affect the global, static, and volatile variables

Since the variable is volatile, the value of a,b,c will be write back into the memory. If setjump execute successfully in the function f1(), the jump point is set and the addresses of local variables are saved.

The memory status after f1():

Address	Value
0x0000000C	103
0x0000000B	102
0x0000000A	101
….	

Fig 1. Memory Stack

In the function f2(), the function will first declare two new variables. The variable f and g will be allocated to the same memory space of b and c respectively[1]. (because the property of variable stack). The change will be written back to memory since variable is also volatile. Thus, (a,b,c) is now (101,201,202) in memory stack

Address	Value
0x0000000C	202
0x0000000B	201
0x0000000A	101
….	

Fig 2. Memory Stack

After the longjmp to f1(), the state is resume. Printf will pint the value in the memory instead of register (1,2,3) which is "101 201 202".

2.

This program avoid zombie processes by calling fork twice. The parent of second child is set to init while the first child is killed and init will reap the status.

[1] System will find two space for the local variables at the same time.

```
int main(void) {
    pid_t pid;
    /*fork first child*/
    if ( (pid = fork()) < 0){
        err_sys("fork error");
    }else if ( pid == 0) {
        /*first child*/
        if ( (pid = fork()) < 0) { /*fork second child*/
            err_sys("fork error");
        }else if( pid > 0)exit(0); /*terminate first child*/
        /*second child*/
        sleep(2);
        printf("Second child, parent pid = %d\n",getppid());
        exit(0);
    }
    /*parent will wait for first child*/
    /*parent*/
    if(waitpid(pid, NULL, 0) != pid) err_sys("waitpid
error");
    exit(0);
}
```

Quiz#2

1.

If the signal occurs after the test of sig_int_flag but before the call to pause, the process could go to sleep forever (assuming that the signal is never generated again).

2.

The problem with this code fragment is that there is a window of time—after the signal has occurred, but before the call to signal in the signal handler—when the interrupt signal could occur another time. This second signal would cause the default action to occur, which for this signal terminates the process

Quiz#3

1.

Register handler once and for all and we don’t have to reestablished the handler.

Delivered & Pending : A process has the option of *blocking* the delivery of a signal.

POSIX.1 allows the system to deliver the signal either once or more than once

2.

Setjump: jump across functions

Sigsetjump: the mask will be save and restore when siglongjmp is called

The process's current signal mask is saved in env and will be restored if a siglongjmp(3) is later performed with this env.