System Programming
Assignment #4

Student Name: 林楷恩
Student ID: b07902075

(a) 1 Without zombie processes, parent may not be able to know the the termination status of its child processes if they terminate before the parent `wait` for them. With zombie processes, parent can fetch its child's termination status no matter when the child processes die.

2 Without zombie state, the relationship between parent and children may miss. For example, a parent process can end up trying to wait for the return code of another process's child.

(b) Because `fork()` make the child have a independent memory space containing the same data as its parent at the time of `fork()`. However, the child may not change some of the memory and just read it or ignore it, which makes it unnecessary to make new copy of it. *copy-on-write* means the system only copys those which are modified by child process, and thus saves the time to copy the other data.

(c)
```
pid_t double_fork()
{
    pid_t pid_child, pid_grandchild;
    int pfd[2];
    pipe(pfd);
    pid_child = vfork();
    if ( pid_child == 0 ) {
        pid_grandchild = fork();
        if ( pid_grandchild == 0 ) {
            // grandchild
            // close unused fd
            close(pfd[1]);
            // read message from parent to ensure child (grandchild's
               parent) is dead
            char msg[128];
            read(pfd[0], msg, 128);
            close(pfd[0]);
            /* do stuff */
            return 0;
        } else {
            // child
            _exit(0);
        }
    } else {
        // parent
        // close unused fd
        close(pfd[0]);
        // wait child process
        waitpid(pid_child, NULL, 0);
        // tell grandchild that child is dead
        char *msg = "The child is dead!";
        write(pfd[1], msg, strlen(msg));
        close(pfd[1]);
        // because vfork() make the child process share
        // the same memory space, the parent's variable
        // also has the pid of the grandchild
        return pid_grandchild;
    }
}
```