Task 01

General requirements

Please submit the assignment as a `*.zip` archive, where every problem is stored in its own subdirectory. All the code must be well-commented and must come with a Makefile which builds a program. Notes and explanations may come in a separate file in the problem's directory. For notes and explanations use text or pdf format.

1 Problem 1

Find out what the following terms mean when speaking about UNIX processes: *zombie*, *orphan*, *daemon*. Answer the following questions providing reasonable explanations:

1. What happens when a child kills its parent?
2. Can a zombie do so, or it is too dead for that?
3. Can a daemon become a zombie?
4. How long can an orphan live on it's own, if at all?

2 Problem 2.1

Write a program in C which reports local time in the system log every 10 seconds.

Use `time.h` functions to get local time and convert it to string, use `syslog.h` to handle logging facilities.

3 Problem 2.2

Make sure that the time-reporting program from the previous problem keeps running after its parent or grandparents terminate. *Hint:* learn about daemons in UNIX.

Hand-out: 15/01/2013 Hand-in: 29/01/2013

4 Problem 3.1

Write a program in C that executes the `ls` command, mirroring command line arguments, reads the output and outputs it on *stdout* only if the number of lines is smaller than *25*. Otherwise the program should state: "too many files".

Use the `fork` function to cerate a subprocess; use the `execv` family of functions to launch `ls` and use pipes to read the output produced by the `ls` command. Command line arguments passed to the `main` function via its second argument (*argv*) could be reused within `execv` function call. The following C code loads the content of the *fd* file descriptor into memory.

```
char* buf = NULL;
char* bufp;
size_t bufsz, cursz, curpos;
ssize_t ssz;
struct stat st;

/* Assuming that FD is a file descriptor opened for
   reading, get file system I/O blocksize into BUFSZ.  */
if (fstat (fd, &st) >= 0) {
        bufsz = (size_t) st.st_blksize;
} else {
        /* Handle error.  */
}

/* Allocate buffer of size BUFSZ.  */
buf = (char *) malloc (bufsz);
curpos = 0;
cursz = bufsz;

/* Block read FD, storing data into BUF.  */
while ((ssz = read (fd, buf + curpos, bufsz)) > 0) {
        curpos += ssz;
        cursz = curpos + bufsz;
        if (NULL == (bufp = (char *) realloc (buf, cursz))) {
                /* Handle error.  */
        }
        buf = bufp;
}

/* Zero-terminate BUF.  */
buf[curpos] = 0;
```

5 Problem 3.2

Modify the previous task and pipe the output of the `ls` command through `less` command if the output is longer than *25* lines.