BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Pada era Industri 4.0, sebagian besar aplikasi desktop dibuat dengan antarmuka grafis (GUI) untuk memudahkan pengguna awam dalam memanfaatkan dan mengoperasikan aplikasi tersebut. Meskipun demikian, para ahli lebih menyukai antarmuka teks (CLI) untuk melakukan perintah-perintah *superuser*, repetitif, *piped*, *redirected*; serta perintah-perintah yang sulit dikerjakan dengan GUI, terutama pada bidang otomasi pengembangan perangkat lunak.[1]–[3] Oleh karena itu, penelitian pada – pemanfaatan CLI dan/atau pemrograman *shell* pada sistem-sistem operasi *UNIX-like* yang mana sering digunakan oleh para ahli; pengembang; dan *system admin* – perlu dilakukan.

UNIX, yang merupakan pendahulu sistem-sistem operasi *UNIX-like* seperti Darwin (macOS); BSD; dan Linux, telah memiliki satu standar/spesifikasi yang disebut dengan Single UNIX Specification (SUS). Standar ini dikembangkan dan dikelola oleh Austin Group, yang merupakan kerja sama para ahli dari IEEE, ISO JTC1 SC22, dan The Open Group.[4], [5] Standar tersebut terus berkembang sesuai dengan perkembangan zaman dan kebutuhan industri. Meskipun demikian, tidak semua sistem operasi *UNIX-like* mampu untuk selalu mengikuti perkembangan standar tersebut.[6]

Pada implementasinya, sistem-sistem operasi *UNIX-like* tersebut menerapkan idealismenya masing-masing ke sistem-sistem operasi mereka walaupun para pengembang sistem pada umumnya menargetkan kesesuaian dengan standar-standar POSIX yang merupakan inti dari SUS.[7] Perbedaan idealisme tersebut berdampak pada perbedaan *environment* pada masing-masing sistem operasi tersebut walaupun secara garis besar sistem-sistem operasi tersebut menyediakan/mengimplementasikan utilitas-utilitas (perintah) inti yang sama seperti `date`, `awk`, `sed`, dan lain sebagainya.[8] Perbedaan implementasi SUS

tersebut menghambat para pengembang dalam membuat dan *deploying* program *shell* yang lintas *platform*.[9]

Sesuai dengan pembahasan sebelumnya, sebuah *wrapper library* untuk menjembatani perbedaan-perbedaan *environment* dan implementasi tersebut diperlukan; karena argumen, variabel *environment*, dan parameter utilitas-utilitas inti di sistem operasi *UNIX-like* satu dengan sistem operasi *UNIX-like* lainnya belum tentu kompatibel. Oleh karena itu, pada penelitian ini, penulis merancang dan membangun sebuah *wrapper library* bersumber terbuka berlisensi MIT untuk Bash dan *shell-shell* populer lainnya yang penulis beri nama "Bridge.sh". *Wrapper library* tersebut menerjemahkan panggilan *shell* dan memungkinkan *shell* melakukan fitur utilitas-utilitas tambahan yang tersedia di sistem lain tetapi tidak tersedia di sistem yang saat ini digunakan.

Penulis meng-*open-source*-kan kode sumber Bridge.sh[10] agar komunitas dapat memberi sumbangsih terbaiknya. Penulis memilih lisensi MIT untuk proyek ini karena lisensi jenis ini sangat permisif dan tidak mengekang seperti GPLv3 – yang memaksa seseorang yang membuat produk turunan proyek berlisensi GPLv3 – untuk menggunakan lisensi GPLv3 juga. Selain itu, lisensi GPLv3 juga tidak membolehkan suatu produk berlisensi GPLv3 untuk di-*bundle* dengan suatu produk *proprietary*. Hal tersebut bertentangan dengan tujuan penulis yang ingin agar *compatibility layer* ini bebas digunakan untuk akademik maupun komersial. [11]

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang di atas, pada penelitian ini, penulis merumuskan bagaimana proses rancang bangun dan implementasi sebuah *wrapper library* yang menjembatani sistem operasi *UNIX-like* satu dengan yang lain. Penulis juga merumuskan analisis dampak dan keberlangsungan dari pengembangan *wrapper library* ini.

1.3. Pertanyaan Penelitian

Berikut adalah beberapa pertanyaan yang penulis ajukan dalam penelitian ini.

1. Bagaimana cara mengimplementasikan Bridge.sh?
2. Apa kendala yang akan dihadapi dalam rancang bangunnya?
3. Apa dampak negatif dan positif dari hasil rancang bangun Bridge.sh?

1.4. Batasan Masalah

Pada penelitian ini, penulis membatasi penelitian pada:

1. Implementasi pencegatan dan translasi panggilan *shell*.
2. Implementasi kekurangan fitur utilitas-utilitas tambahan yang masih termasuk dalam standar SUS/POSIX, seperti `awk`, `cp`, `ls`, dan sebagainya.
3. Implementasi kekurangan – fitur utilitas-utilitas di luar standar SUS/POSIX; seperti `sudo`, `readlink`, `apt`, dan sebagainya – tidak dilakukan.
4. Implementasi *workaround* dan *error handling*.
5. Target *platform shell* penulis adalah Bash dan Zsh.
6. Target sistem operasi penulis adalah Linux (*desktop* dan Android) dan macOS.
7. Target C *standard library* penulis adalah Glibc dan BSD Libc.

1.5. Tujuan Penelitian

Berikut adalah tujuan-tujuan dari penelitian ini.

1. Mengembangkan sebuah *wrapper library* yang menjebatani perbedaan *environment* antar sistem-sistem operasi *UNIX-like*.
2. Mengetahui perbedaan-perbedaan *environment* antar sistem-sistem operasi *UNIX-like*.
3. Mengetahui perbedaan penggunaan, fitur-fitur, dan cara kerja *shell-shell* pada sistem operasi *UNIX-like*.

1.6. Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

1. Menjebatani perbedaan *environment* antar sistem-sistem operasi *UNIX-like.*
2. Memungkinkan skrip yang dibuat untuk sistem operasi *UNIX-like* satu berjalan dengan baik pada sistem operasi *UNIX-like* lainnya.
3. Memudahkan pengguna dalam beralih dari sistem operasi *UNIX-like* satu ke sistem operasi *UNIX-like* lainnya.
4. Memungkinkan *single codebase approach* karena pengguna tidak harus menulis perintah berbeda untuk *platform* lain.
5. Membuka banyak kemungkinan lainnya pada penelitian-penelitian selanjutnya.