

SKRIPSI

PERKAKAS COMMAND LINE KIRI



Alfred Aprianto Liaunardi

NPM: 6181801014

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022**

DAFTAR ISI

DAFTAR ISI	iii
DAFTAR GAMBAR	v
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	3
2 LANDASAN TEORI	5
2.1 <i>Command Line</i>	5
2.1.1 <i>Command Line Interface</i> dan <i>Graphical User Interface</i>	5
2.1.2 <i>Command Line</i> di Linux	6
2.1.3 <i>Command Line</i> di Windows	7
2.2 KIRI	10
2.2.1 Tampilan	11
2.2.2 API	11
DAFTAR REFERENSI	15
A KODE PROGRAM	17
B HASIL EKSPERIMEN	19

DAFTAR GAMBAR

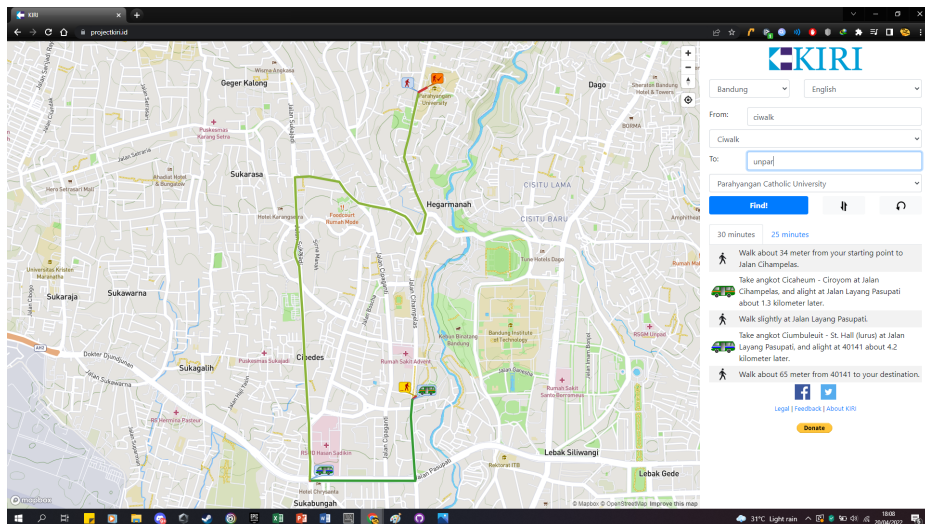
1.1	Tampilan halaman web KIRI	1
2.1	Dua jenis tampilan perangkat lunak	5
2.2	Baris <i>shell prompt</i> terminal di sistem operasi Linux.	6
2.3	Tampang kedua antarmuka <i>command line</i> bawaan di sistem operasi Windows.	8
2.4	Tampilan awal halaman web KIRI	11
2.5	Tampilan akhir halaman web KIRI	12
B.1	Hasil 1	19
B.2	Hasil 2	19
B.3	Hasil 3	19
B.4	Hasil 4	19

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Project KIRI¹ (akan disingkat sebagai KIRI dalam dokumen ini) adalah sebuah perangkat lunak berbasis web yang dibuat untuk membantu mengurangi efek dari kemacetan. KIRI mengurangi dampak kemacetan dengan membantu pengguna, baik masyarakat maupun turis, dalam menggunakan salah satu sarana transportasi umum yang ada di Indonesia, yaitu angkutan kota (angkot). Cara KIRI mempermudah penggunaan angkot adalah dengan menunjukkan rute yang akan ditempuh, beserta langkah-langkah yang harus dilakukan oleh pengguna yang ingin berpergian dari satu titik ke titik lain, mulai dari seberapa jauh pengguna harus berjalan untuk menaiki angkot yang bersangkutan, di mana pengguna harus naik atau turun, seberapa jauh lagi pengguna harus berjalan sampai ke titik tujuan, dan seberapa lama estimasi waktu perjalanan yang akan ditempuh. Untuk kebutuhan pembuatan perangkat lunak yang memanfaatkan fitur dari KIRI, tersedia juga REST API KIRI yang dapat digunakan secara praktis. Adapun tampilan dari halaman web ini dapat dilihat di gambar 1.1.



Gambar 1.1: Tampilan halaman web KIRI, yang menunjukkan rute dari Cihampelas Walk ke Universitas Katolik Parahyangan.

Sementara itu, dalam komputer, salah satu dari sekian banyak tipe perangkat lunak adalah *command line*. *Command line* adalah perangkat lunak paling sederhana, yang sudah ada sejak

¹<https://projectkiri.id>

pertama kali komputer diciptakan. Perangkat lunak selalu memiliki tampilan berupa *command line interface* (CLI), yang tidak memiliki tampilan apapun selain sebuah kotak yang memuat teks berupa perintah-perintah tertentu, baik perintah yang meminta masukan dari user untuk dilakukan oleh komputer, maupun perintah yang menampilkan keluaran dari komputer, tanpa ada tambahan gambar grafis apapun, seperti pada perangkat lunak dengan tampilan *graphical user interface* (GUI). Singkatnya, tipe perangkat lunak ini bukan merupakan tipe yang paling indah untuk dilihat oleh para pengguna, tetapi jika digunakan dengan tepat, maka jenis perangkat lunak ini bisa menyuruh komputer untuk melakukan banyak sekali perintah-perintah dengan sangat cepat dan sangat efektif.

Pada skripsi ini akan dibuat sebuah perangkat lunak berupa perkakas *command line* (*command line tool*) yang dapat menjalankan fungsi-fungsi API dari KIRI. Perangkat lunak ini, seperti jenisnya, akan dibuat murni sebagai perkakas yang dijalankan dari *command line* (terminal, cmd, PowerShell, dll.), dan tampilan akhir dari perangkat lunak akan berupa *command line interface* tanpa tambahan *graphical user interface*. Keseluruhan dari perangkat lunak ini akan dibangun dalam bahasa C.

1.2 Rumusan Masalah

1. Bagaimana membangun perkakas *command line* yang dapat mengimplementasikan fitur-fitur API KIRI dalam bahasa C?
2. Bagaimana integrasi perkakas *command line* KIRI dapat dilakukan dengan perkakas-perkakas *command line* lainnya di Linux?

1.3 Tujuan

Batasan masalah dalam skripsi ini adalah sebagai berikut:

1. Membangun perkakas *command line* yang dapat mengimplementasikan fitur-fitur API KIRI dalam bahasa C.
2. Melakukan integrasi perkakas *command line* KIRI dengan perkakas-perkakas *command line* lainnya di Linux.

1.4 Batasan Masalah

Batasan masalah dalam skripsi ini adalah sebagai berikut:

1. Perangkat lunak dibuat murni dalam bentuk CLI, tanpa tambahan GUI.
2. Perangkat lunak yang dibuat tidak menyelesaikan batasan (lokasi tidak terdeteksi, rute tidak berhasil ditemukan, dsb.) yang sudah sejak awal terdapat dalam KIRI.

1.5 Metodologi

Metodologi yang akan diikuti dalam skripsi ini adalah sebagai berikut:

1. Melakukan studi dan eksplorasi terhadap fungsi-fungsi yang dimiliki perangkat lunak KIRI serta cara implementasi API KIRI.
2. Melakukan analisis dan desain perangkat lunak yang akan dibangun.

3. Melakukan studi dan eksplorasi terhadap seluruh kemungkinan *library-library* yang memenuhi spesifikasi dalam pembuatan perangkat lunak, berdasarkan analisis dan desain yang telah dilakukan sebelumnya.
4. Melakukan analisis kebutuhan fitur-fitur perangkat lunak dan melakukan eksplorasi *library* yang dapat digunakan dan memenuhi spesifikasi dalam pembuatan perangkat lunak.
5. Membangun perangkat lunak berdasarkan rancangan yang sudah dibuat, dengan mengimplementasikan seluruh modul dan *library* yang telah ditentukan di tahap sebelumnya dalam bahasa C.
6. Melakukan pengujian fungsional, perbaikan *bug*, serta rekomendasi perbaikan berdasarkan pengujian yang sudah dilakukan.
7. Menyelesaikan pembuatan dokumen-dokumen yang berkaitan, seperti dokumen skripsi dan dokumentasi perangkat lunak.

1.6 Sistematika Pembahasan

Setiap bab dalam skripsi ini memiliki sistematika pembahasan dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.

2. Bab 2: Dasar Teori

Bab ini berisi pembahasan-pembahasan teoritis mengenai aspek-aspek yang akan dirujuk di dalam skripsi ini, seperti *command line*, bahasa C, dan juga KIRI.

3. Bab 3: Analisis dan Perancangan

Bab ini berisi pembahasan mengenai rancangan perangkat lunak serta seluruh analisis yang dilakukan terhadap kebutuhan fitur perangkat lunak.

4. Bab 4: Implementasi dan Pengujian

Bab ini berisi pembahasan mengenai pembuatan perangkat lunak, implementasi seluruh modul-modul yang telah ditentukan di bab 3, serta pengujian fitur-fitur dari perangkat lunak.

5. Bab 5: Kesimpulan dan Saran

Bab ini berisi kesimpulan hasil pembuatan perangkat lunak dan saran-saran terhadap hasil perangkat lunak yang diberikan selama pengerjaan skripsi.

BAB 2

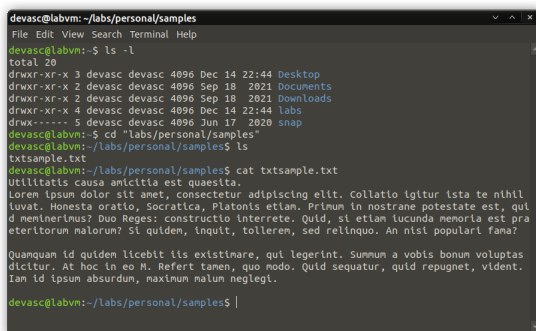
LANDASAN TEORI

2.1 *Command Line*

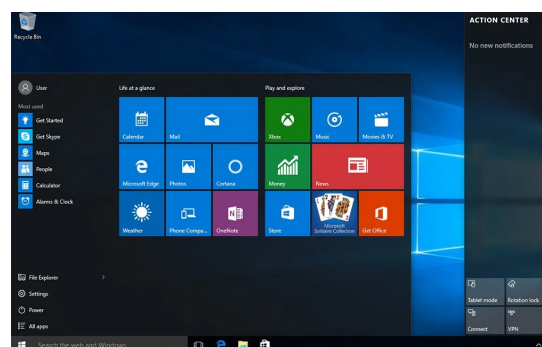
Command line (atau *command line interface*) dapat diartikan sebagai tampilan antarmuka/*interface* yang memproses perintah dari pengguna dan meneruskannya langsung ke sistem operasi untuk dijalankan.[1] Seluruh sistem operasi komputer yang ada memiliki sebuah *command line interface* dalam bentuk *shell*, yang dapat digunakan oleh penggunanya untuk langsung mengakses fungsi atau servis yang disediakan oleh sistem operasi.[2]

2.1.1 *Command Line Interface dan Graphical User Interface*

Ada beberapa dari tipe antarmuka yang masih banyak digunakan di zaman sekarang, tetapi dua tipe yang paling banyak muncul adalah *command line interface* dan *graphical user interface*. Perangkat lunak berbasis *command line* sendiri bisa memiliki berbagai macam tampilan, tetapi semuanya selalu mengikuti satu bentuk antarmuka umum. Bentuk yang dimaksud adalah sebuah area/*window* yang memuat teks berupa perintah-perintah dari user untuk dilakukan oleh komputer, beserta keluarannya yang juga berupa teks, seperti dapat dilihat pada gambar 2.1a. Jenis perangkat lunak seperti ini disebut memiliki antarmuka jenis *command line interface* (CLI). Adapun dekorasi visual yang dimiliki oleh jenis tampilan ini hanya berupa warna pada teks-teks yang ada, tanpa tambahan gambar apapun. Jika perangkat lunak tersebut memiliki dekorasi dan/atau tombol interaktif berupa gambar grafis, seperti pada gambar 2.1b, maka perangkat lunak tersebut dikategorikan sebagai perangkat lunak berbasis *graphical user interface*.



(a) Antarmuka perangkat lunak berbasis *command line interface*.



(b) Antarmuka perangkat lunak berbasis *graphical user interface*.

Gambar 2.1: Contoh dua jenis antarmuka (*interface*) perangkat lunak.

Selain dari tampilannya sendiri, ada beberapa perbedaan utama lain antara perangkat-perangkat lunak berbasis *command line interface* dengan perangkat lunak berbasis *graphical user interface*. Adapun perbedaan-perbedaan utama dari kedua jenis antarmuka ini adalah sebagai berikut.

- Penggunaan sumber daya sistem untuk menjalankan perangkat lunak berbasis *command line interface* lebih rendah dibandingkan dengan perangkat lunak berbasis *graphical user interface*.
- Bagi pengguna pemula (atau pengguna awam pada umumnya), perangkat lunak berbasis *command line interface* akan lebih sulit digunakan karena tidak adanya bantuan apapun dalam bentuk visual, sehingga satu-satunya cara untuk tahu bagaimana cara menggunakan fitur-fiturnya adalah melalui dokumentasi perangkat lunak yang ada. Karena alasan yang sama pula, perangkat lunak berbasis *command line interface* lebih sulit untuk dibiasakan penggunaannya.
- Automasi perintah yang bersifat berulang-ulang jauh lebih mudah dilakukan pada perangkat lunak berbasis *command line interface*. Hal ini dikarenakan perangkat lunak berbasis *command line interface* tidak hanya lebih mudah untuk dibuat *script*-nya, tetapi juga lebih efisien untuk digunakan ketika ada banyak sekali perintah yang harus dilakukan pada suatu saat tertentu.[2]

2.1.2 *Command Line* di Linux

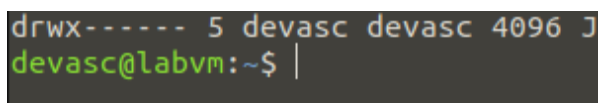
Linux merupakan sebuah sistem operasi yang sangat modular, jadi ada banyak sekali *shell* yang dapat dijalankan dan digunakan di dalamnya. Walaupun begitu, ada satu *shell* yang selalu datang ter-*install* di dalam semua sistem operasi Linux, yaitu "*bash*" (GNU *Bourne Again Shell*).[3]

Tampilan

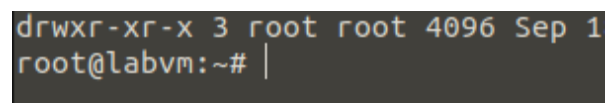
Ketika terminal di Linux dijalankan, akan keluar kotak dialog, beserta sebuah baris. Baris ini biasanya berisi sebuah teks dengan format sebagai berikut.

```
<nama pengguna>@<nama perangkat>:<direktori yang sedang diproses>$
```

Tanda dolar di ujung baris ini menandakan bahwa baris tersebut merupakan baris *shell prompt*, yang merupakan waktu di mana terminal sudah siap menerima masukan dari pengguna untuk diproses. Perlu diingat bahwa di posisi tanda dolar ini, terkadang justru terdapat tanda pagar (#). Tanda pagar di akhir baris *shell prompt* menandakan bahwa terminal tersebut dijalankan dengan tingkat akses *superuser*, yang berarti bahwa entah pengguna masuk ke sistem sebagai user *root*, atau terminal memiliki izin tingkat *superuser/administrator*. [1]



(a) *Shell prompt* terminal dengan tingkat izin normal.



(b) *Shell prompt* terminal dengan tingkat izin *superuser*.

Gambar 2.2: Baris *shell prompt* terminal di sistem operasi Linux.

1 Navigasi

2 Sama seperti di Windows, Linux menyimpan file-filenya di sebuah struktur direktori yang bersifat
3 hierarkial. Hal ini berarti bahwa file-file tersebut disimpan dalam direktori-direktori (atau *folder-*
4 *folder*) yang tersusun seperti sebuah pohon. dalam arti bahwa satu *folder* bisa jadi berada di dalam
5 satu *folder* lain, atau berisi beberapa *folder* lainnya.[1]

6 Untuk navigasi, terminal Linux memiliki beberapa perintah utama. Adapun perintah-perintah
7 tersebut adalah sebagai berikut.

- 8 • **pwd** [1]

9 **pwd** merupakan singkatan dari *print working directory*, yang berarti bahwa perintah ini akan
10 mengeluarkan *working directory*, atau direktori tempat terminal sekarang sedang bekerja/ber-
11 jalan, sebagai keluaran dari perintah tersebut. Ketika pengguna pertama kali menjalankan
12 terminal, *working directory*-nya selalu merupakan direktori *home* dari perangkat.

- 13 • **ls** [1]

14 **ls** digunakan untuk menghasilkan keluaran berupa isi dari folder yang dispesifikasi. Biasanya
15 digunakan ketika pengguna sudah memasuki folder yang diinginkan, walaupun dengan perintah
16 ini, pengguna bisa saja mengintip isi dari folder manapun di direktori manapun, dengan
17 mengikutkan direktori yang diinginkan sebagai parameter dari perintah tersebut. Adapun
18 Isi dari folder yang diikutkan sebagai parameter tidak hanya berupa folder lain, tetapi juga
19 seluruh file-file yang ada, walaupun untuk file-file yang disembunyikan (nama file diawali
20 dengan tanda titik), perlu ditambahkan opsi **-a** agar file-file tersebut muncul pula dalam
21 keluarannya.

- 22 • **cd** [1]

23 **cd** adalah perintah yang berfungsi untuk mengganti *working directory* dari terminal. Untuk
24 melakukan hal tersebut, perintah yang perlu dimasukkan adalah sebagai berikut:

25 **cd <direktori yang diinginkan>**

26 Direktori yang diinginkan dapat berupa direktori absolut, atau direktori relatif. Perbedaanannya
27 adalah direktori absolut selalu dimulai dari folder *root*, mengikuti folder-folder apapun yang
28 ada di antara *root* sampai ke folder yang diinginkan.

29 Sedangkan, direktori relatif selalu dimulai dari *working directory*. Untuk penggunaan direktori
30 relatif, diperlukan dua buah notasi spesial, yaitu titik (**.**), yang merepresentasikan *working*
31 *directory* sekarang itu sendiri, dan dua titik (**..**), yang merepresentasikan *parent folder* dari
32 *working directory*.

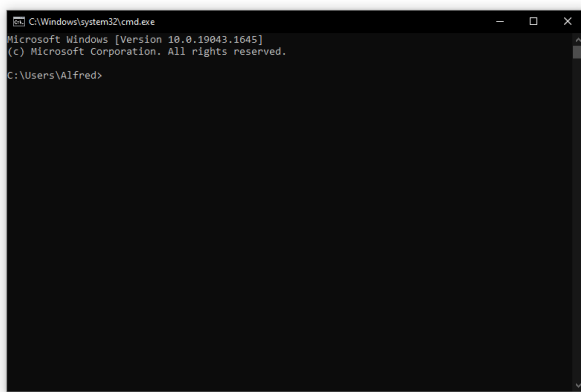
33 2.1.3 Command Line di Windows

34 Cara kerja *command line* di Windows serupa dengan cara kerja *command line* di Linux, dalam
35 arti bahwa untuk bekerja dengan *command line* di Windows, penggunaanya juga akan langsung
36 berinteraksi dengan utilitas yang disediakan oleh sistem operasi. *Command line* di Windows juga
37 dapat digunakan untuk hal-hal yang serupa dengan *command line* di Linux, seperti menulis (dan
38 menjalankan) *script*, menjalankan perintah yang diinginkan pengguna secara otomatis, atau melihat
39 status dari sistem operasi.[2]

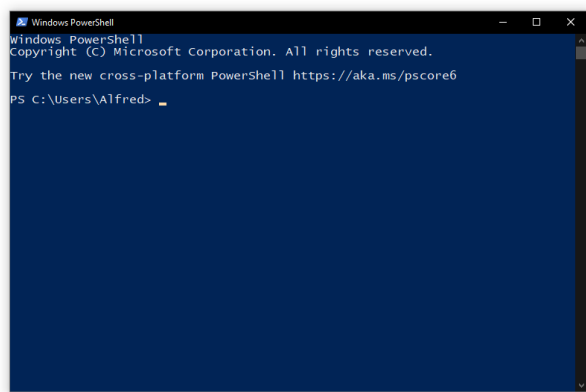
Masih sama dengan Linux, ada banyak sekali command yang bisa digunakan, sehingga susah untuk menghafal seluruh command-command yang ada—termasuk masukan, parameter-parameter yang dibutuhkan, serta keluarannya. Untuk melihat dokumentasi, atau penjelasan detail untuk masukan, keluaran, parameter, serta opsi-opsi dari perintah tertentu, pengguna dapat memasukkan perintah tersebut, diikuti dengan `/?`.^[2]

Tampilan

Di sistem operasi Windows, ada dua jenis antarmuka *command line*, yaitu `cmd` (*Command Prompt*) dan *PowerShell*. Keduanya memiliki tampilan yang kurang lebih sama—hanya saja awalnya `cmd` memiliki latar belakang hitam, sedangkan *PowerShell* memiliki latar belakang biru tua, seperti terlihat di gambar 2.3.



(a) Antarmuka Windows *Command Prompt* (`cmd`)



(b) Antarmuka Windows *PowerShell*

Gambar 2.3: Tampilan kedua antarmuka *command line* bawaan di sistem operasi Windows.

Navigasi

Untuk navigasi di antarmuka *command line* Windows, ada dua perintah penting yang dipakai ketika pengguna sedang berurusan dengan file-file dan navigasi dalam direktori sistem. Kedua perintah tersebut adalah `cd` dan `dir`.

- `cd (chdir)`¹

`cd` merupakan sebuah perintah yang memiliki tiga fungsi utama, yaitu menampilkan *drive* tempat sedang *command line* berada (jika pengguna hanya memasukkan `cd` tanpa parameter apapun), menampilkan direktori tempat *command line* sedang berada (jika pengguna hanya memasukkan *drive* sebagai parameter, atau fungsi yang paling umumnya, untuk mengganti *working directory* dari *command line*.

Adapun format dari perintah `dir` adalah sebagai berikut.

```
cd [/d] [<drive>:][<path>]
```

Dengan fungsi dari semua opsi dan parameter yang ada sebagai berikut.

¹[cd | Microsoft Docs](#)

1 – /d

2 Opsi yang menandakan bahwa pengguna ingin mengganti *drive* (partisi) dan juga *working*
3 *directory* dari *command line*.

4 – <drive>:

5 Kode huruf dari partisi yang akan diproses.

6 – <path>

7 Direktori yang akan diproses. Parameter ini harus diikuti beserta kode huruf partisi
8 (tidak dapat berdiri sendiri.)

9 • **dir**

10 **dir** merupakan sebuah perintah yang mengeluarkan/menampilkan sebuah daftar berisi file-file
11 yang ada di suatu direktori, termasuk subdirektori. Jika tidak disertai parameter apapun,
12 perintah ini akan menampilkan label volume dan nomor serial *disk*, dilanjutkan dengan daftar
13 direktori dan file di dalamnya. Untuk file, akan ditampilkan nama beserta ukurannya. Perintah
14 ini juga akan menampilkan jumlah direktori dan file yang didaftarkan, ukuran kumulatifnya,
15 dan sisa dari *disk* yang tidak terpakai (dalam *bytes*).²

16 Adapun format dari perintah **dir** adalah sebagai berikut.^[2]

17 **dir** [drive:] [path] [filename] [/A[:]attributes] [/B] [/C] [/D] [/L] [/N]
18 [/O[:]sortorder] [/P] [/Q] [/R] [/S] [/T[:]timefield] [/W] [/X] [/4]

19 Untuk perintah ini, seperti terlihat di atas, memiliki banyak sekali opsi dan parameter.
20 Tiap-tiap dari parameter tersebut memiliki fungsi tersendiri, yaitu:

21 – /A[:]attributes

22 Menampilkan file-file dengan atribut tertentu, seperti file yang disembunyikan, file sistem,
23 file *read-only*, dan sebagainya.

24 – /B

25 Menghilangkan *heading* dan ringkasan informasi dari keluaran, atau dengan kata lain,
26 hanya menampilkan file-file dan direktori, tanpa informasi tambahan apapun.

27 – /C

28 Menggunakan separator koma untuk tiap angka ribuan di ukuran file. Jika opsi yang
29 dimasukkan adalah /-C, separator koma justru akan dihilangkan.

30 – /D

31 Menampilkan keluaran dengan format yang lebih lebar. Jika opsi ini diikuti, keluaran
32 akan ditampilkan dengan urutan berdasarkan kolom.

33 – /L

34 Seluruh teks dalam keluaran akan menggunakan huruf kecil. Jika opsi ini tidak digunakan,
35 keluaran akan mengandung huruf besar dan huruf kecil *mixed case*.

36 – /N

37 Menampilkan daftar dengan format panjang, dengan nama file berada di ujung paling
38 kanan.

39 – /O[:]sortorder

40 Menampilkan daftar direktori yang terurut berdasarkan urutan tertentu, seperti ber-
41 dasarkan ekstensi file, berdasarkan tanggal dibuat, berdasarkan nama, dan sebagainya.

²[dir | Microsoft Docs](#)

Jika tidak diikuti tanda minus (-) sebelum huruf O pada perintah, daftar yang muncul akan terurut secara menaik.

– /P

Sementara memberhentikan keluaran (memberi jeda kecil) setelah setiap halaman informasi.

– /Q

Menambahkan informasi mengenai pemilik file dalam keluaran.

– /R

Menampilkan *data stream* alternatif, jika ada.

– /S

Mendaftarkan seluruh file di direktori dan subdirektori yang diproses. Tiap-tiap direktori akan memiliki *header* tersendiri dalam keluarannya.

– /T[:,timefield]

Menspesifikasi *time field* mana yang akan tampil dan digunakan sebagai urutan, jika aturan pengurutan lain tidak ditentukan. *Time field* yang dapat digunakan adalah waktu pembuatan file, kapan terakhir file diakses, dan kapan file terakhir dimodifikasi. Jika parameter ini tidak dispesifikasi, *time field* yang digunakan adalah kapan file terakhir dimodifikasi.

– /W

Menampilkan keluaran dengan format yang lebih lebar. Opsi ini hampir sama dengan /D, hanya saja untuk /W, jika opsi ini diikuti, keluaran akan ditampilkan dengan urutan berdasarkan baris, dan bukan kolom.

– /X

Menampilkan nama pendek yang dibuat untuk nama-nama file non-8.3. Format penampilannya akan sama seperti opsi /N, dengan nama pendeknya ditampilkan di keluaran sebelum nama panjangnya.

– 4

Menampilkan angka tahun sebagai angka empat digit.

2.2 KIRI

KIRI merupakan sebuah perangkat lunak berbasis web yang berfungsi untuk menyelesaikan (atau setidaknya mengurangi) dampak dari masalah-masalah yang dapat diselesaikan oleh transportasi umum/publik di Indonesia, seperti pemanasan global, kemacetan, atau peningkatan harga bensin. Selain itu, turis mancanegara juga memilih untuk menaiki transportasi umum, karena sarana transportasi tersebut tidak hanya jauh lebih murah, tetapi juga memberikan kesempatan kepada mereka untuk melihat seluk-beluk dari kota-kota yang mereka kunjungi. Walaupun begitu, masih banyak masyarakat lokal sendiri yang segan untuk menaiki transportasi publik, umumnya karena transportasi publik lebih rumit persiapannya dibandingkan dengan transportasi privat, seperti kendaraan pribadi.³

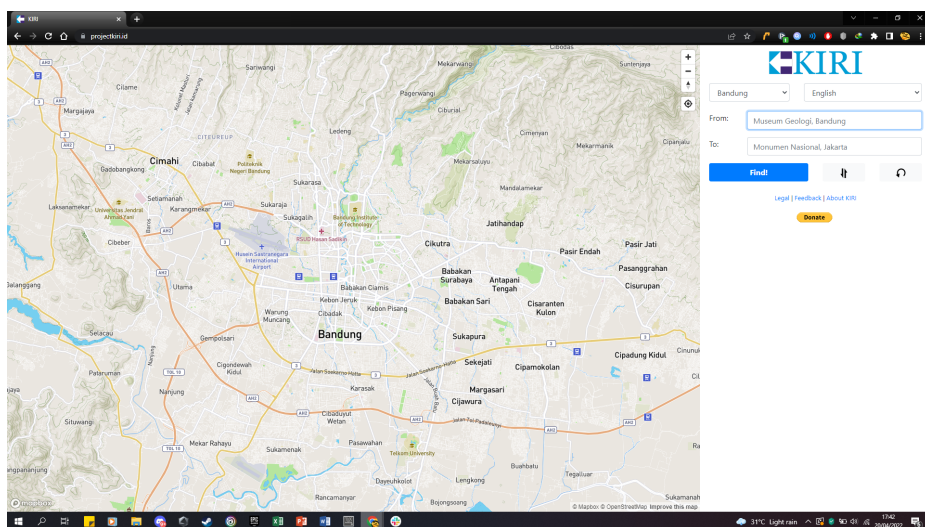
Di halaman web KIRI, pengguna dapat memasukkan input berupa lokasi awal dan lokasi tujuan

³<https://projectkiri.github.io/#about-kiri>

1 dan KIRI akan menghasilkan seluruh langkah yang harus ditempuh oleh pengguna untuk sampai
2 ke lokasi tujuan, dengan menggunakan angkot. Keluaran ini sudah meliputi kode angkot mana saja
3 yang harus dinaiki, dan juga seberapa jauh pengguna harus berjalan kaki untuk sampai ke lokasi
4 rute angkot berikutnya.

5 2.2.1 Tampilan

6 Pada saat pertama kali dibuka, hal pertama yang paling mencolok di halaman awal web KIRI adalah
7 sebuah peta besar di sebelah kiri yang dapat diperbesar ataupun diperkecil. Sedangkan, bagian
8 kanan dari halamannya terdiri atas beberapa bagian. Di bagian paling atas terdapat logo KIRI,
9 beserta sepasang menu *dropdown* - yang pertama merupakan pilihan kota tempat pengguna berada
10 (untuk sekarang hanya tersedia pilihan kota Jakarta dan Bandung), dan yang kedua merupakan
11 pilihan bahasa, entah bahasa Indonesia atau Inggris. Di bawahnya merupakan sepasang menu
12 *dropdown* yang merupakan tempat di mana pengguna memasukkan lokasi awal dan tujuan yang
13 akan diproses oleh KIRI. Terakhir, di bawahnya ada sebuah bagian kosong, yang nantinya akan
14 menjadi tempat di mana KIRI akan meletakkan keluaran dari prosesnya. Adapun tampilan awal
15 dari halaman web ini dapat dilihat di gambar 2.4.

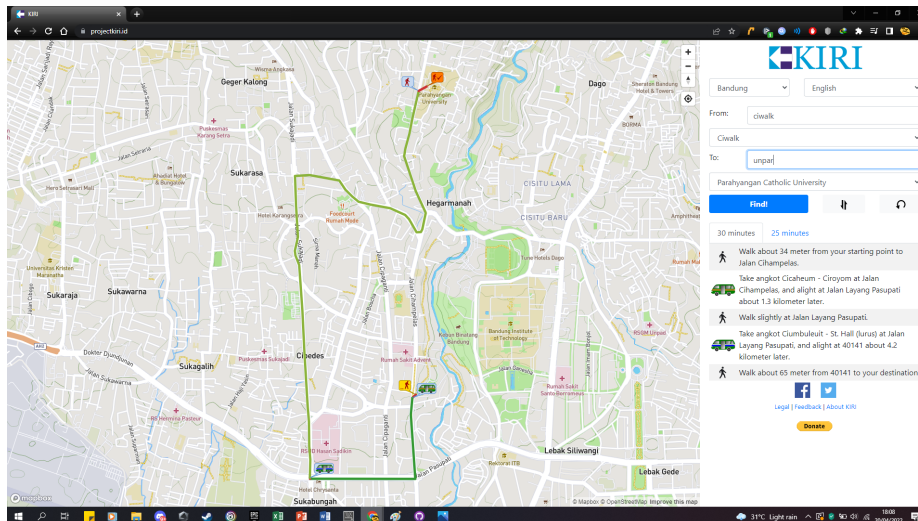


Gambar 2.4: Tampilan awal halaman web KIRI.

16 Ada dua area yang memiliki perbedaan yang signifikan ketika pengguna sudah memasukkan
17 masukan dan menyuruh KIRI untuk memprosesnya. Bagian yang pertama adalah bagian peta, yang
18 setelah pemrosesan masukan, akan memiliki garis-garis berwarna yang menandakan rute angkot
19 maupun tujuan perjalanan kaki yang harus ditempuh oleh pengguna. Bagian kedua adalah bagian
20 keluaran, yang tadinya kosong, sekarang akan berisi langkah-langkah yang harus ditempuh oleh
21 pengguna untuk pergi dari lokasi awal ke lokasi tujuan. Spesifiknya, perbedaan-perbedaan ini
22 dapat dilihat di gambar 2.5.

23 2.2.2 API

24 KIRI juga memiliki sebuah API yang dapat digunakan untuk keperluan pengembangan perangkat
25 lunak. Seluruh permintaan (*request*) yang dilakukan melalui API KIRI harus dilakukan sebagai



Gambar 2.5: Tampilan halaman web KIRI setelah pemrosesan masukan dari pengguna selesai.

- 1 permintaan tipe GET ke <https://projectkiri.id/api>, beserta parameter-parameter yang dibutuhkan.
- 2 Permintaan tersebut harus memiliki parameter-parameter seperti terlihat di bawah ini.⁴
- 3
 - 4 • **version**
 - 5 **Kemungkinan nilai:** 2
 - 6 Parameter ini merupakan tanda bagi API untuk menggunakan protokol versi 2.
 - 7 • **mode**
 - 8 **Kemungkinan nilai:** findroute
 - 9 Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna.
 - 10 Untuk mode **findroute**, jasa yang akan digunakan adalah jasa pencarian rute dengan angkot.
 - 11 • **locale**
 - 12 **Kemungkinan nilai:** en atau id
 - 13 Parameter ini mengatur bahasa apa yang akan digunakan dalam keluaran API nantinya—**en**
 - 14 berarti keluaran akan menggunakan bahasa Inggris, dan **id** berarti keluaran akan menggunakan
 - 15 bahasa Indonesia.
 - 16 • **start**
 - 17 **Kemungkinan nilai:** lat, lng; dalam bentuk desimal
 - 18 Parameter ini merupakan nilai *latitude* dan *longitude* dari titik awal perjalanan pengguna.
 - 19 • **finish**
 - 20 **Kemungkinan nilai:** lat, lng; dalam bentuk desimal
 - 21 Parameter ini merupakan nilai *latitude* dan *longitude* dari titik akhir/tujuan perjalanan
 - 22 pengguna.
 - 23 • **presentation**
 - 24 **Kemungkinan nilai:** desktop
 - 25 Parameter ini hanya digunakan untuk fitur *backwards compatibility*.
 - 26 • **apikey**
 - 27 **Kemungkinan nilai:** angka heksadesimal 16-digit
 - Parameter ini berisi kunci API pribadi yang harus digenerasi terlebih dahulu sebelum API

⁴<https://github.com/projectkiri/Tirtayasa/wiki/KIRI-API-v2>

dapat digunakan.

Sedangkan, respon yang diberikan oleh API berupa sebuah objek JSON yang selalu memiliki setidaknya dua variabel, yaitu:⁵

- **status**

Kemungkinan nilai: `ok` atau `error`

Variabel ini manandakan apakah permintaan berhasil diproses atau tidak. Jika permintaan berhasil diproses, variabel ini akan bernilai `ok`, dan jika tidak, variabel ini akan bernilai `error`.

- **message**

Variabel ini bisa berisi dua macam objek. Jika permintaan dari user tidak berhasil diproses, atau dalam kata lain, terjadi sebuah `error`, maka variabel ini akan berisi string yang merupakan pesan `error` serta alasan spesifik mengapa `error` tersebut terjadi. Di lain sisi, jika permintaan dari user berhasil diproses, variabel ini akan mengalami dua perubahan utama. Pertama, nama variabel ini akan berubah menjadi `routingresults`, dan kedua, isi dari variabel ini akan menjadi sebuah `array` JSON yang merupakan respon dari API KIRI berupa keluaran yang akan dilihat oleh pengguna. `Array` JSON ini sendiri terbagi menjadi beberapa variabel lainnya, yang dapat dilihat di daftar di bawah ini.

- **steps**

Tipe: `array`

Variabel ini merepresentasikan satu buah langkah yang harus ditempuh oleh pengguna. Adapun `array` ini sendiri berisi variabel-variabel berikut:

- * Tipe transportasi

Tipe sarana transportasi yang harus dipakai oleh pengguna. Jika pengguna harus berjalan kaki, variabel ini akan berisi `walk`. Jika pengguna harus menaiki angkot, variabel ini akan berisi `angkot`.

- * Kode angkot

Variabel ini menunjukkan angkot mana yang harus dinaiki oleh pengguna di langkah tersebut. Jika penggunaan angkot tidak dimungkinkan pada langkah ini (pengguna harus berjalan kaki), variabel ini akan berisi `walk`.

- * Array *latitude* dan *longitude* lokasi

`Array` nilai-nilai desimal *latitude* dan *longitude* dari berbagai titik lokasi yang terdapat dalam rute.

- * Deskripsi langkah

Deskripsi langkah yang harus ditempuh, dalam bahasa natural. Bahasa apa yang digunakan untuk deskripsi ini tergantung parameter `locale` yang diatur dalam masukan.

- * URL untuk mendapatkan tiket kendaraan

Tautan untuk mendapatkan tiket angkutan umum, jika diperlukan. Jika transportasi pada langkah tersebut tidak memerlukan tiket, variabel ini akan berisi `null`.

- * URL editor rute

Tautan untuk meng-edit rute, jika situasinya memungkinkan. Jika tidak, variabel ini akan berisi `null`.

⁵<https://github.com/projectkiri/Tirtayasa/wiki/KIRI-API-v2>

- 1 – `traveltime`
- 2 **Tipe:** string
- 3 Variabel ini berisi estimasi jangka waktu yang diperlukan untuk menyelesaikan langkah
- 4 tersebut.

DAFTAR REFERENSI

- [1] Shotts Jr., W. E. (2019) *The Linux Command Line*, 5th internet edition. <https://www.linuxcommand.org/tlcl.php>.
- [2] Mueller, J. P. (2007) *Windows® Administration at the Command Line for Windows Vista™, Windows® 2003, Windows® XP, and Windows® 2000*, 1st edition. Wiley Publishing, Inc., Indiana.
- [3] Neil Matthew, R. S. (2007) *Beginning Linux® Programming*, 4th edition. Wiley Publishing, Inc., Indiana.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf

```

Kode A.2: MyCode.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }

```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4