Perkakas Command Line KIRI

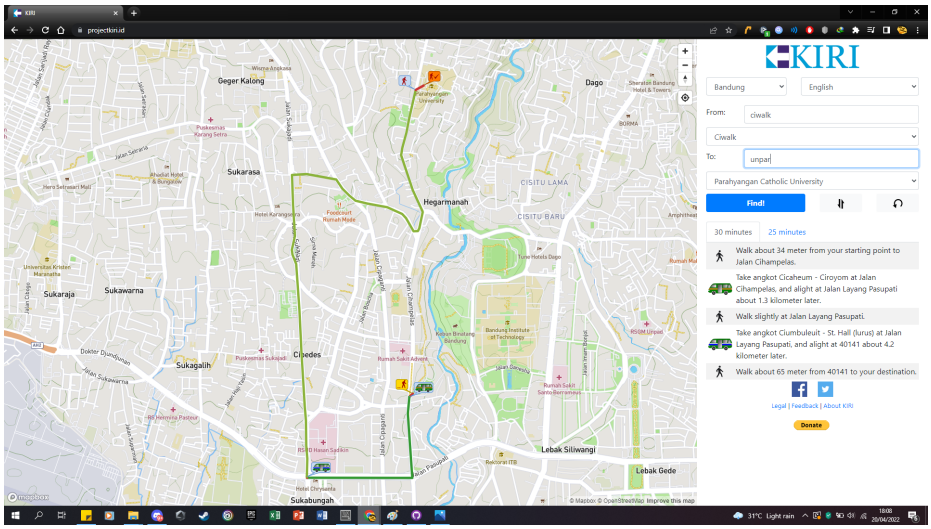
Alfred Aprianto Liaunardi–6181801014

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian Nugroho, M.Comp.**
Pembimbing pendamping: -
Kode Topik : **PAN5201**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **52 - Genap 21/22**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Latar Belakang

Project KIRI[1] (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 penggunanya, 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.



Gambar 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* (*command line interpreter*, atau *command line interface*) adalah sebuah perangkat lunak

[1] https://projectkiri.id

berupa sebuah kotak/*window* yang memuat teks berupa perintah-perintah,[2] yang menerima masukan dari pengguna dan menjalankannya.[1] Perintah-perintah ini hanya berupa gabungan dari teks and simbol-simbol berupa karakter, tanpa ada tambahan gambar grafis apapun. 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.

3 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?

4 Tujuan

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.

5 Deskripsi Perangkat Lunak

Perangkat lunak akhir yang akan dibuat memiliki fitur minimal sebagai berikut:

- Pengguna dapat memasukkan lokasi awal dan tujuan akhir sebagai masukan dari perangkat lunak.
- Pengguna dapat melihat langkah-langkah yang harus ditempuh dalam perjalanan, mulai dari angkot mana saja yang harus dinaiki, ke mana pengguna harus berjalan kaki untuk bisa mencapai angkot terdekat dari lokasi terakhir pengguna, sampai seberapa jauh pengguna harus berjalan untuk mencapai tujuan akhir.
- Pengguna dapat melihat jarak yang harus ditempuh untuk setiap langkahnya.
- Pengguna dapat melihat seberapa lama waktu perjalanan untuk setiap langkahnya.

6 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terkahir :

1. **Melakukan eksplorasi fungsi-fungsi perangkat lunak KIRI serta eksplorasi cara implementasi API KIRI.**

[2] Ubuntu Tutorials - The Linux command line for beginners: 3. Opening a Terminal

Status : Ada sejak rencana kerja skripsi.
Hasil : 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 jenis sarana transportasi tersebut tidak hanya jauh lebih murah, tetapi juga memberikan kesempatan yang mudah kepada mereka untuk melihat seluk-beluk dari kota-kota yang mereka kunjungi. Walaupun begitu, banyak masyarakat lokal sendiri yang seringkali masih segan untuk menaiki transportasi publik, umumnya karena transportasi publik dianggap lebih rumit persiapannya dibandingkan dengan metode-metode transportasi privat, seperti menaiki kendaraan pribadi.[3]

Di halaman web KIRI, pengguna dapat memasukkan input berupa lokasi awal dan lokasi tujuan dan KIRI akan menghasilkan seluruh langkah yang harus ditempuh oleh pengguna untuk sampai ke lokasi tujuan, dengan menggunakan angkot. Keluaran ini sudah meliputi kode angkot mana saja yang harus dinaiki, dan juga seberapa jauh pengguna harus berjalan kaki untuk sampai ke lokasi rute angkot berikutnya.

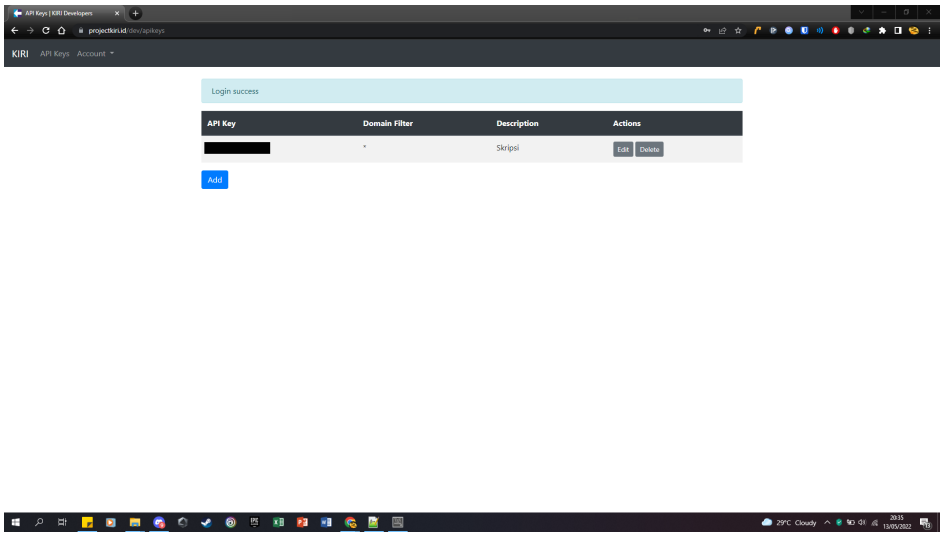
KIRI juga memiliki sebuah API yang dapat digunakan untuk keperluan pengembangan perangkat lunak. API ini menyediakan tiga buah jenis layanan web (*webservice*), yang ketiganya dapat dilakukan dengan mengirim pemintaan (*request*) tipe GET melalui API tersebut. Isi dari permintaan yang perlu dikirimkan serta respon dari API yang akan dikembalikan berbeda tergantung dari jenis layanan yang digunakan. Adapun ketiga jenis layanan tersebut adalah pencarian tempat (*search place*), pencarian rute (*routing*), dan *smart direction*.

Search Place

Layanan pencarian lokasi (*search place*) adalah layanan web pada API KIRI yang berfungsi untuk mencari suatu lokasi berdasarkan kata kunci yang diberikan oleh pengguna. Untuk menggunakan layanan ini, pengguna harus mengirim permintaan GET ke alamat https://projectkiri.id/api. Adapun permintaan tersebut harus memiliki parameter-parameter seperti terlihat di bawah ini.

- `version`
  Kemungkinan nilai: 2
  Parameter ini merupakan tanda bagi API untuk menggunakan protokol versi 2.
- `mode`
  Kemungkinan nilai: `searchplace`
  Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna. Untuk pengunaan layanan pencarian lokasi, variabel ini harus diisi dengan `searchplace`.
- `region`
  Kemungkinan nilai: `cgk`, `bdo`, `mlg`, atau `sub`
  Parameter ini merupakan kode bandara IATA tiga huruf yang merepresentasikan daerah mana tempat lokasi yang ingin dicari berada. Kode yang dapat diproses oleh API ini meliputi `cgk` (Cengkareng/Jakarta), `bdo` (Bandung), `mlg` (Malang), dan `sub` (Surabaya).

[3] https://projectkiri.github.io/#about-kiri



Gambar 2: Halaman web *API Keys* KIRI.

- `querystring`
  Kemungkinan nilai: *string* apapun dengan panjang minimal satu karakter
  Parameter ini berisi kata kunci yang akan digunakan untuk menentukan lokasi yang ingin dicari pengguna.
- `apikey`
  Kemungkinan nilai: angka heksadesimal 16-digit
  Parameter ini berisi kunci API pribadi yang harus digenerasi terlebih dahulu sebelum API dapat digunakan.

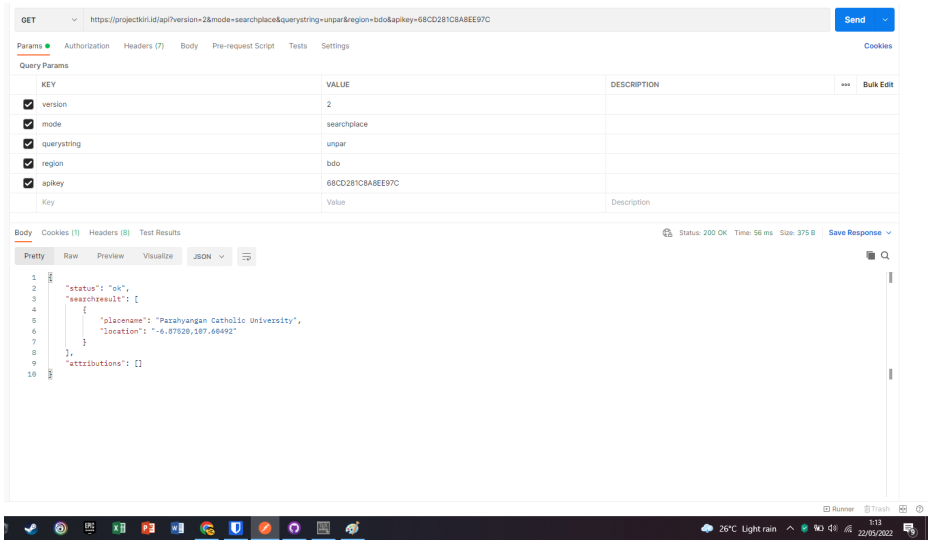
Perlu diperhatikan bahwa salah satu dari parameter yang harus diikutkan dalam pesan tersebut merupakan parameter yang meminta kunci API. Kunci tersebut harus digenerasikan terlebih dahulu sebelum API KIRI dapat digunakan, melalui halaman *API Keys* KIRI,[4] yang dapat dilihat di gambar 2.

Untuk mengakses halaman tersebut, pengguna harus membuat sebuah akun terlebih dahulu. Ketika akun sudah dibuat, maka pengguna baru akan dapat membuat kunci API yang dibutuhkan, sekaligus membuat filter *domain*, yang membatasi di *domain* mana saja kunci tersebut dapat digunakan, serta memberikan deskripsi untuk kunci API tersebut. Kunci ini kemudian dapat digunakan sebagai nilai dari parameter `apikey` yang diperlukan dalam permintaan tadi.

Sebelum membahas keluaran dari layanan API ini, perlu ditegaskan dulu apa definisi dari nilai *latitude* dan *longitude* suatu lokasi. *Latitude* merupakan berapa derajat sebuah tempat berada dari garis ekuator, dengan lokasi-lokasi yang berada maksimum 90 derajat di atas ekuator memiliki nilai *latitude* positif, sedangkan lokasi-lokasi yang berada maksimum 90 derajat di bawah ekuator memiliki nilai *latitude* negatif. Sedangkan, *longitude* merupakan berapa derajat lokasi sebuah tempat berada dari garis meridian (bujur) utama Bumi, dengan rentang nilai dari -180 derajat di sisi kiri (barat) meridian utama, hingga 180 derajat di kanan (timur) bujur tersebut.[5] Kedua nilai ini merupakan salah satu dari dua variabel yang dikembalikan dalam respon API untuk layanan ini, dengan variabel lainnya berupa nama dari lokasi yang ditemukan itu sendiri. Adapun respon yang diberikan oleh API akan berupa sebuah objek JSON yang selalu memiliki setidaknya dua variabel, yaitu:

- `status`

[4] https://projectkiri.id/dev/apikeys
[5] https://gsp.humboldt.edu/olm/Lessons/GIS/01%20SphericalCoordinates/Latitude_and_Longitude.html



Gambar 3: Penggunaan API KIRI untuk layanan pencarian lokasi menggunakan Postman. Gambar ini menunjukkan hasil pencarian lokasi "unpar" di daerah Bandung.

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`.

- `searchresult`
  Variabel ini merupakan hasil pencarian yang ditemukan oleh layanan API ini. Isi dari variabel ini merupakan objek *array* yang masing-masing memiliki variabel berikut:
  - `placename`
    Variabel ini berisi nama lokasi yang ditemukan berdasarkan kata kunci yang diberikan oleh pengguna.
  - `location`
    Variabel ini berisi nilai *latitude* dan *longitude* dari lokasi yang ditemukan dalam pencarian.

Contoh dari penggunaan API KIRI untuk layanan ini dapat dilihat di gambar 3.

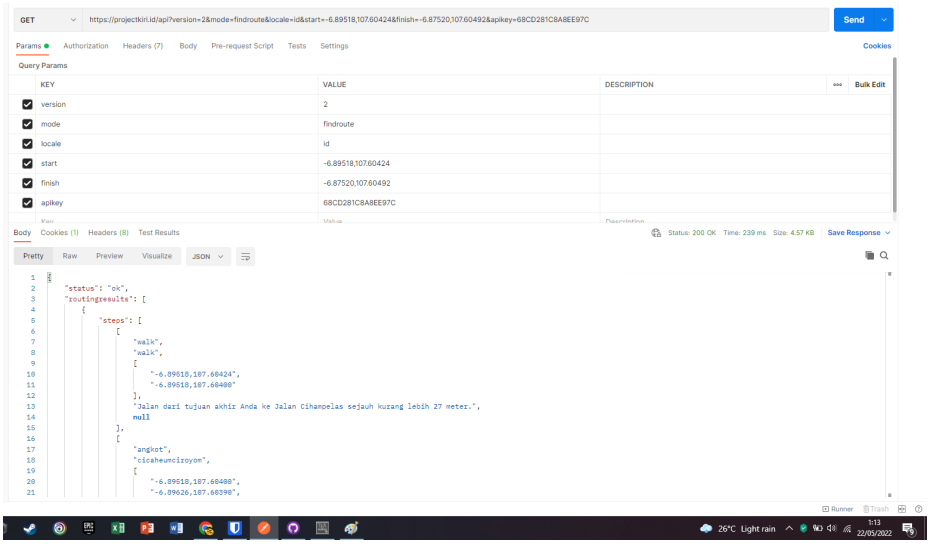
Routing

Layanan pencarian rute (*routing*) adalah layanan web pada API KIRI yang memiliki fungsi yang sama dengan fungsi utama dari perangkat lunak KIRI sendiri, yaitu menunjukkan rute serta langkah-langkah yang harus ditempuh untuk pergi dari satu lokasi ke lokasi lainnya, dengan menggunakan angkot yang tersedia. Untuk menggunakan layanan ini, pengguna harus mengirim permintaan GET ke alamat https://projectkiri.id/api. Adapun permintaan tersebut harus memiliki parameter-parameter seperti terlihat di bawah ini.

- `version`
  Kemungkinan nilai: `2`
  Parameter ini merupakan tanda bagi API untuk menggunakan protokol versi 2.
- `mode`
  Kemungkinan nilai: `findroute`
  Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna. Untuk pengunaan layanan pencarian rute dengan angkot, variabel ini diisi dengan `findroute`.
- `locale`
  Kemungkinan nilai: `en` atau `id`
  Parameter ini mengatur bahasa apa yang akan digunakan dalam keluaran API nantinya—`en` berarti keluaran akan menggunakan bahasa Inggris, dan `id` berarti keluaran akan menggunakan bahasa Indonesia.
- `start`
  Kemungkinan nilai: `lat`, `lng`; dalam bentuk desimal
  Parameter ini merupakan nilai *latitude* dan *longitude* dari titik awal perjalanan pengguna.
- `finish`
  Kemungkinan nilai: `lat`, `lng`; dalam bentuk desimal
  Parameter ini berisi nilai *latitude* dan *longitude* dari titik akhir/tujuan perjalanan pengguna.
- `presentation` (opsional)
  Kemungkinan nilai: `desktop`
  Parameter ini hanya digunakan untuk fitur *backwards compatibility*.
- `apikey`
  Kemungkinan nilai: angka heksadesimal 16-digit
  Parameter ini berisi kunci API pribadi yang harus digenerasi terlebih dahulu sebelum API dapat digunakan.

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

- `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:



Gambar 4: Penggunaan API KIRI untuk layanan pencarian rute menggunakan Postman. Gambar ini menunjukkan hasil pencarian rute dari Cihampelas Walk ke UNPAR.

* 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 yang digunakan 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 melakukan modifikasi rute, jika dimungkinkan. Jika rute tidak bisa dimodifikasi, variabel ini akan berisi `null`.

– `traveltime`
  Tipe: string
  Variabel ini berisi estimasi jangka waktu yang diperlukan untuk menyelesaikan langkah tersebut.

Adapun gambar 4 menunjukkan penggunaan API KIRI untuk layanan pencarian rute dari Cihampelas Walk ke Universitas Katolik Parahyangan.

Smart Direction

Layanan terakhir dari API ini adalah layanan *smart direction*, yang merupakan gabungan dari kedua

layanan sebelumnya. Berbeda dengan kedua layanan tadi, yang harus mengakses API secara manual (dengan mengirimkan permintaan GET), layanan ini tidak memerlukan pengguna untuk mengirim permintaan tersebut secara manual—layanan ini sudah otomatis mengirimkan permintaan tersebut. Berbeda dengan kedua layanan sebelumnya pula, layanan ini tidak memerlukan dibuatnya kunci API terlebih dahulu.

Layanan ini bekerja dengan menyuruh peramban pengguna untuk langsung mengakses halaman web KIRI yang sudah langsung menunjukkan rute yang perlu ditempuh untuk pergi dari lokasi satu ke lokasi lainnya. Untuk melakukan hal ini, layanan ini memerlukan sebuah URL, yang memiliki format sebagai berikut:

```
https://projectkiri.id?start=<lokasi awal>&finish=<lokasi akhir>&locale=<locale>
```

Dapat dilihat bahwa URL tersebut memiliki tiga buah parameter, yaitu:

- `start`
  Kemungkinan nilai: Nilai *latitude* dan *longitude* lokasi, atau nama lokasi tersebut
  Parameter ini berisi lokasi yang ingin digunakan sebagai lokasi mulainya pencarian rute.
- `finish`
  Kemungkinan nilai: Nilai *latitude* dan *longitude* lokasi, atau nama lokasi tersebut
  Parameter ini berisi lokasi yang merupakan tujuan akhir yang ingin dicapai dalam pencarian rute.
- `locale` (opsional)
  Kemungkinan nilai: `id` atau `en`
  Menentukan dalam bahasa apa hasil pencarian rutenya akan ditampilkan (bahasa Indonesia atau bahasa Inggris). Jika parameter ini tidak diberikan oleh pengguna, maka bahasa yang akan digunakan adalah bahasa yang terakhir dipakai di halaman web KIRI sendiri.

Misalkan pengguna ingin mencari rute dari Cihampelas Walk ke Universitas Katolik Parahyangan, dan menampilkan langkah-langkah yang harus ditempuh dalam rutenya dalam bahasa Indonesia. Pengguna dapat memasukkan URL berikut ke peramban mereka.

```
https://projectkiri.id?start=ciwalk&finish=unpar&locale=id
```

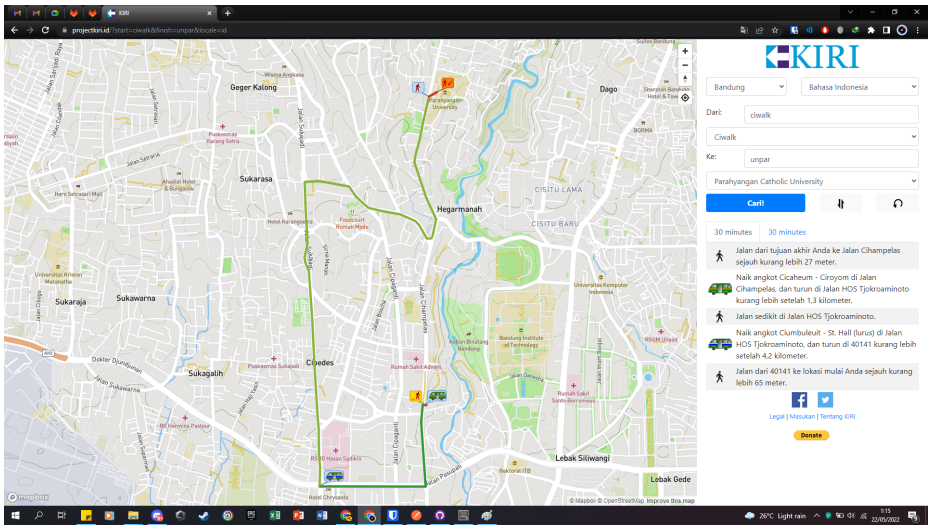
Jika URL tersebut sudah dimasukkan ke kotak *link* pada peramban, halaman yang akan ditampilkan akan terlihat seperti pada gambar 5.

2. **Mempelajari perkakas-perkakas *command line* sejenis.**
   Status : Ditambahkan pada Skripsi 1.
   Hasil : Perkakas-perkakas sejenis yang dianalisis meliputi *Chrome Web Store Item Property CLI* dan *iTunes Search API*.

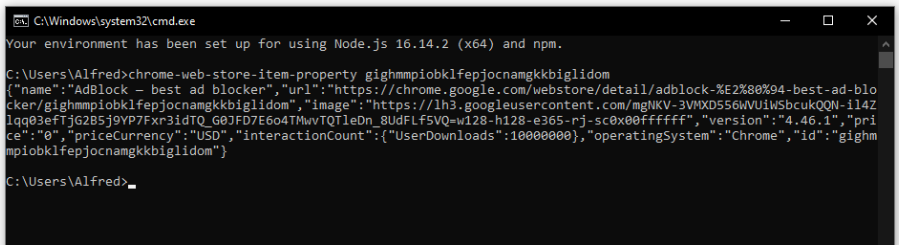
   - *Chrome Web Store Item Property CLI*
     Perkakas ini dapat digunakan melalui *command prompt* dengan cara mengetikkan perintah sebagai berikut.

     ```
     chrome-web-store-item-property <identifier>
     ```

     Dengan `identifier` berupa ID dari ekstensi yang diinginkan. Jadi, misalkan pengguna memasukkan `gighmmpiobklfepjocnamgkkbiglidom` sebagai ID yang akan digunakan sebagai *identifier*,



Gambar 5: Penggunaan API KIRI untuk layanan *smart direction*, dari Cihampelas Walk ke UNPAR.



Gambar 6: Contoh penggunaan perkakas *Chrome Web Store Item Property CLI*.

maka perkakas ini akan mengembalikan metadata dari ekstensi "AdBlock" sebagai keluarannya. Contoh penggunaan perkakas ini dapat dilihat di gambar 6.

Sedangkan, keluaran dari perkakas ini merupakan sebuah objek JSON dengan properti-properti sebagai berikut.

- `name`
  Nama dari ekstensi yang dicari metadatanya.
- `url`
  URL halaman web dari ekstensi yang dicari di *web store* Google Chrome.
- `image`
  Logo (dan ikon *thumbnail*) dari ekstensi yang dicari metadatanya.
- `version`
  Nomor versi dari ekstensi.
- `price`
  Harga dari ekstensi. Jika ekstensi tidak memiliki harga yang perlu dibayarkan (gratis), properti ini akan bernilai `0`.
- `priceCurrency`
  Kode mata uang dari harga ekstensi. Jika ekstensi tidak memiliki harga yang perlu dibayarkan, properti ini akan berisi "`USD`".
- `interactionCount`
  Properti ini berisi interaksi-interaksi pengguna yang tercatat sebagai data di halaman *web store* ekstensi. Pada saat pembuatan skripsi ini, properti ini hanya memiliki satu buah subproperti, yaitu `userDownloads`, yang menandakan berapa kali ekstensi ini telah diunduh

oleh pengguna di manapun.

- `operatingSystems`
  Menandakan di peramban mana ekstensi versi ini dapat diinstal. Karena ekstensi-ekstensinya berada di *web store* Chrome,
- `ratingValue` (tidak digunakan lagi)
  Peringkat yang diberikan oleh para pengguna ekstensi ini. Nilai dari properti ini berupa skala desimal dari 0.00 sampai dengan 5.00. Di versi terbaru dari perkakas ini, properti ini tidak lagi tersedia dalam keluarannya.
- `ratingCount` (tidak digunakan lagi)
  Jumlah pengguna yang telah menilai/memberi peringkat ke ekstensi ini. Di versi terbaru dari perkakas ini, properti ini tidak lagi tersedia dalam keluarannya.
- `id`
  Properti ini mengandung ID dari ekstensi tersebut. Nilai dari properti ini akan sama dengan ID yang digunakan sebagai parameter masukan perkakas.

- *iTunes Search API*
  Perkakas *command line* ini berfungsi untuk melakukan pencarian melalui API iTunes, sehingga seakan-akan pengguna langsung melakukan pencarian di iTunes sendiri. Hasil pencarian yang dilakukan termasuk judul lagu, nama artis, ataupun nama album, dan pengguna dapat memilih secara spesifik objek apa yang ingin dicari.

  Perkakas ini dapat digunakan melalui *command prompt* dengan cara mengetikkan perintah sebagai berikut.

  ```
  itunes-search-api <input> [<options>]
  ```

  Dengan `input` berupa nama dari objek yang dicari. Perkakas ini juga memiliki opsi yang masing-masing memiliki parameter tersendiri untuk mempersempit hasil pencarian. Adapun opsi-opsi tersebut dapat dilihat di daftar di bawah ini.

  - `country`
    Kemungkinan nilai: Kode negara dua huruf
    Opsi ini menerima parameter berupa kode negara asal dari album atau artis yang dicari.
  - `entity`
    Kemungkinan nilai: `song`, `musicArtist`, atau `album`
    Menandakan jenis objek/entitas yang ingin dicari. Opsi ini dapat bernilai `song` untuk pencarian berbasis judul lagu, `musicArtist` untuk pencarian nama artis, atau `album` untuk pencarian nama album. Jika opsi ini tidak dipakai, objek apapun yang memiliki kemiripan dengan `input` dalam salah satu dari ketiga properti ini akan muncul dalam hasil pencarian.
  - `limit`
    Kemungkinan nilai: Bilangan bulat positif[6]
    Jumlah hasil pencarian maksimal yang ingin ditampilkan dalam keluaran.

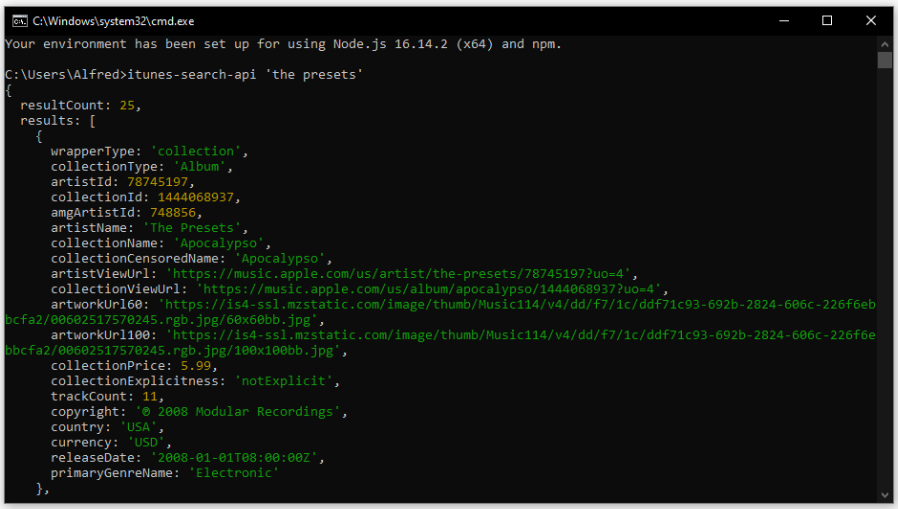
  Sedangkan, keluaran dari perkakas ini merupakan sebuah objek JSON yang memiliki dua properti utama, yaitu:

  - `resultCount`
    Properti ini berisi bilangan bulat yang menandakan berapa buah objek yang terdapat dalam hasil pencarian.

[6] Opsi ini juga menerima bilangan bulat negatif, tetapi menggunakan sebuah bilangan bulat negatif akan menghilangkan pengaruh opsi ini terhadap hasil keluaran.

– `results`

Array yang berisi kumpulan objek yang terdapat di dalam hasil pencarian. Objek-objek ini akan dikembalikan berupa sebuah *array* lain yang berisi seluruh properti dari masing-masing objek. Apa saja properti yang diikutkan dalam *array* tersebut tergantung tipe dari objek dalam hasil pencarian.



Gambar 7: Contoh penggunaan perkakas *iTunes Search API*. Gambar hanya memuat satu objek untuk menghemat tempat.

Adapun contoh penggunaan dan hasil keluaran perkakas ini dapat dilihat di gambar 7.

3. **Mempelajari bahasa pemrograman C serta mempelajari dokumentasi-dokumentasi dari seluruh modul yang dibutuhkan untuk pembuatan perangkat lunak.**
Status : Ada sejak rencana kerja skripsi.
Hasil : Ada beberapa modul/fungsi bawaan dari bahasa C yang akan digunakan dalam pembuatan perkakas *command line* ini. Salah satu dari fungsi tersebut adalah fungsi `getopt`

getopt [2]

`getopt` merupakan sebuah fungsi yang dapat mengautomasi pekerjaan-pekerjaan yang berhubungan dengan penerimaan opsi-opsi untuk *command line* berbasis UNIX.

Fungsi `getopt` dapat dipanggil dengan format sebagai berikut.

```
getopt (argc, argv, <options>)
```

Seluruh kode ini dapat dimasukkan ke suatu variabel berupa sebuah karakter yang merepresentasikan opsi yang ingin digunakan. `argc` merupakan jumlah argumen yang terdapat dalam masukan, sedangkan argv merupakan sebuah *array* yang berisi argumen-argumen tersebut.

Selain itu, penggunaan `getopt` juga memerlukan penggunaan variabel-variabel tertentu, yang dapat dilihat di daftar berikut.[7]

- `opterr`
Isi dari variabel ini akan memberi signal ke perangkat lunak/perkakas yang menentukan apakah

[7] https://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html

`getopt` akan mengirim pesan ke *error stream* atau tidak. Jika variabel ini bukan bernilai 0, maka pesan *error* akan dikirim. Sebaliknya, jika variabel ini bernilai 0, `getopt` tidak akan mengirim pesan *error* apapun, tetapi tetap akan mengembalikan sebuah karakter tanda tanya (`?`) sebagai tanda bahwa sebuah *error* telah terjadi.

- `optopt`
  Ketika `getopt` menemukan sebuah karakter yang tidak didefinisikan dalam kumpulan opsi, atau sebuah opsi yang tidak disertai argumen yang diperlukan, karakter tersebut akan disimpan di variabel ini.
- `optind`
  Variabel ini digunakan oleh `getopt` sebagai indeks untuk *array* `argv`. Jika seluruh argumen sudah diproses, nilai variabel ini dapat digunakan untuk menentukan argumen mana yang merupakan arguman tambahan yang tidak terpakai. Nilai dari variabel ini dimulai dari 1.
- `optarg`
  Jika opsi yang sedang diproses memerlukan argumen, variabel ini adalah tempat dimana argumen tersebut akan disimpan.
- `<options>`
  Variabel ini berupa *string* yang menandakan karakter-karakter apa saja yang menjadi opsi yang mungkin dalam perkakas tersebut, beserta tipenya. Jika karakter opsi:
  - Diikuti dengan titik dua (`:`), maka opsi tersebut memiliki argumen yang bersifat wajib.
  - Diikuti dengan titik dua ganda (`::`), maka opsi tersebut memiliki argumen yang bersifat opsional.
  - Tidak diikuti apa-apa, maka opsi tersebut merupakan opsi tidak berarguman.

getopt-long

Ada pula versi `getopt` yang memungkinkan perangkat lunak untuk menerima dua jenis opsi—opsi versi pendek berupa sebuah karakter singular, seperti pada `getopt` biasa, dan/atau opsi panjang bergaya GNU, berupa sebuah kata.

`getopt-long` juga memiliki seluruh variabel-variabel yang dimiliki oleh `getopt`, hanya saja `getopt-long` memiliki sebuah variabel tambahan berupa struktur, yaitu `long_options`. Variabel ini merupakan sebuah struktur berupa *array* yang berisi beberapa *array* lainnya, di mana *array-array* lain in merupakan masing-masing opsi dari fungsi `getopt-long` tersebut. Tiap-tiap *array* tersebut memiliki variabel-variabel berikut:

- `name`
  Variabel ini merupakan nama panjang dari opsi.
- `has_arg`
  Variabel ini merupakan penanda apakah opsi memerlukan argumen atau tidak. Nilai yang mungkin dalam variabel ini adalah `no_argument`, `required_argument`, atau `optional_argument`.
- `flag` & `val`
  Kedua variabel ini menandakan bagaimana sebuah opsi akan diberlakukan ketika diterima oleh `getopt-long`. Variabel `flag` dapat diisi dengan penunjuk (*pointer*) ke suatu variabel lain yang akan diisi dengan isi dari variabel `val` untuk menandakan bahwa `getopt-long` telah berhasil memroses opsi tersebut. Di lain sisi, jika variabel ini berisi *null pointer*, maka fungsi `getopt-long` akan mengembalikan isi dari variabel `val`.

Struktur ini harus diakhiri dengan sebuah *array* tambahan yang seluruh variabelnya bernilai 0.

4. **Melakukan analisis dan desain perangkat lunak yang akan dibangun.**
   Status : Ada sejak rencana kerja skripsi—desain akan dilakukan di Skripsi 2.
   Hasil : Pada skripsi ini, akan dibangun sebuah aplikasi berupa perkakas *command line* yang merupakan ekstensi dari sebuah aplikasi berbasis web lain, yaitu KIRI. Perkakas ini akan menjadi ekstensi KIRI dengan cara memungkinkan penggunanya untuk mengakses fungsi-fungsi API KIRI dari *command line* milik perangkat mereka masing-masing. Fungsi utama dari perkakas ini tentunya sama dengan KIRI sendiri, yaitu membantu navigasi dari satu lokasi ke lokasi lain dengan menggunakan angkot.

   Walaupun begitu, aplikasi ini akan memiliki dua fungsi utama yang berhubungan satu sama lain, yaitu pencarian lokasi, dan penunjukkan rute dengan angkot.

   - Pencarian lokasi
     Pencarian lokasi merupakan fungsi utama pertama dari perkakas ini. Untuk fungsi ini, masukan langsung dari pengguna yang akan diterima oleh perkakas adalah kata kunci dari sebuah lokasi yang ingin dicari. Kemudian, perkakas akan memprosesnya melalui API KIRI, lalu mengembalikan nama lokasi tersebut, serta nilai *latitude* dan *longitude*-nya, sebagai keluaran akhir dari proses tersebut.
   - Pencarian rute
     Pencarian rute dengan angkot merupakan fungsi utama kedua, sekaligus fungsi umum dari perkakas ini. Dalam fungsi ini, perkakas akan meminta masukan langsung pengguna berupa nilai *latitude* dan *longitude* dari lokasi awal serta lokasi akhir dari perjalanan pengguna. Setelah masukan ini berhasil diterima dan diproses, perkakas akan mengeluarkan keluaran akhir berupa langkah-langkah yang harus diambil oleh pengguna untuk pergi dari lokasi awal ke lokasi akhir, dengan memanfaatkan jasa angkot yang ada.

   Berikut merupakan opsi-opsi yang akan disediakan dalam perkakas yang akan dibangun.

   - `--help`
     Perlu diingat juga bahwa aplikasi ini merupakan aplikasi *command line* murni, yang berarti bahwa seluruh operasi dari aplikasi ini akan dilakukan tanpa bantuan gambar grafis apapun. Untuk membantu penggunanya dalam mengetahui bagaimana cara menggunakan perkakas ini, tentunya perintah untuk menunjukkan apa perintah-perintah dan opsi-opsi yang tersedia merupakan sebuah keharusan. Hal ini merupakan fungsi satu-satunya dari perintah `--help` ini.
   - `--search`
     Opsi ini akan menandakan bahwa pengguna ingin menggunakan fitur *search place* dari perkakas. Untuk mode ini, perkakas akan menerima input dari pengguna dalam bentuk parameter-parameter dari opsi-opsi tambahan. Adapun opsi-opsi tersebut adalah sebagai berikut.
     - `--region <parameter>`
       Opsi ini merupakan opsi yang akan menerima parameter berupa kota dari lokasi yang ingin dicari.
     - `--querystring <parameter>`
       Opsi ini merupakan opsi yang akan menerima sebuah *string* sebagai parameternya. *String* ini akan digunakan oleh perkakas sebagai kata kunci untuk pencarian lokasi yang ingin ditemukan oleh pengguna.
   - `--route`
     Opsi ini akan menandakan bahwa pengguna ingin menggunakan fitur *find route* dari perkakas. Sama seperti mode `--search`, perkakas akan menerima input dari pengguna dari parameter-parameter milik opsi-opsi tambahan. Adapun opsi-opsi tersebut adalah sebagai berikut.

- `--start <parameter>`
  Opsi ini merupakan opsi yang akan menerima parameter berupa lokasi awal perjalanan pengguna nantinya. Perlu diingat bahwa opsi ini hanya menerima masukan lokasi berupa nilai *latitude* dan *longitude* dari lokasi tersebut.
- `--finish <parameter>`
  Opsi ini merupakan opsi yang akan menerima parameter berupa lokasi akhir perjalanan pengguna nantinya. Sama seperti opsi sebelumnya, parameter ini juga hanya menerima masukan lokasi berupa nilai *latitude* dan *longitude*.
- `--locale <parameter>`
  Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh perkakas di keluarannya nanti.

5. **Melakukan analisis kebutuhan fitur-fitur perangkat lunak.**
   Status : Dipisah dua bersama dengan nomor 6 di Skripsi 1—tidak dikerjakan.
   Hasil : Perangkat lunak hasil skripsi ini merupakan ekstensi dari perangkat lunak lainnya, yaitu KIRI. Oleh karena itu, fitur-fitur yang dimiliki oleh perangkat lunak ini akan sama seperti fitur-fitur yang disediakan oleh KIRI.

6. **Melakukan eksplorasi *library* yang dapat digunakan dan memenuhi spesifikasi dalam pembuatan perangkat lunak.**
   Status : Dipisah dua bersama dengan nomor 5 di Skripsi 1.
   Hasil : Adapun *library-library* yang dapat digunakan berdasarkan spesifikasi perangkat lunak adalah sebagai berikut.

   libcurl [3]

   libcurl merupakan sebuah *library* yang berisi fungsi-fungsi yang disediakan dalam bentuk API bahasa C, untuk digunakan oleh aplikasi-aplikasi bahasa C. Libcurl didesain dengan berorientasi transfer (biasanya transfer berkas), tanpa memerlukan para penggunanya untuk mengerti protokol-protokol yang digunakan dalam proses pentransferan tersebut. Fitur transfer ini dapat dibuat sesederhana mungkin, dan seluruh aturan dan opsi seputar pemindahan berkas tersebut dapat diatur nantinya secara manual melalui opsi-opsi yang ada.

   Untuk mulai melakukan transfer berkas, diperlukan juga sebuah *handle* yang perlu diinisialisasi terlebih dahulu. Adapun cURL memiliki dua jenis *handle*, yaitu *easy handle* dan *multi handle*.

   Easy Handle

   Easy handle dari cURL dapat diinisialisasi dengan memanggil fungsi `curl_easy_init()`. Setelah itu, untuk mengatur opsi-opsi yang perlu diatur sesuai kebutuhan pengguna, seperti URL yang dituju, protokol yang ingin dipakai, koneksi ke port spesifik, dan sebagainya,[8] pengguna harus mengaturnya dengan fungsi `curl_easy_setopt()`. *Handle* ini berhasil diatur opsinya apabila fungsi `curl_easy_setopt()` tadi mengembalikan `CURLE_OK`. Terakhir, untuk menjalankan transfernya, fungsi yang perlu dipanggil adalah `curl_easy_perform(<easy handle>)`, dengan variabel `<easy handle>` diisi dengan nama dari *easy handle* yang ingin dimulai transfernya.

   Handle yang telah diatur ini dapat digunakan berulang kali dengan konfigurasi yang sama, sampai entah pengguna mengganti konfigurasi opsi-opsinya kembali, atau atau *handle*-nya direset dengan pemanggilan fungsi `curl_easy_reset()`.

   Multi Handle

[8] https://curl.se/libcurl/c/curl_easy_setopt.html

Multi handle merupakan sebuah *handle* yang dapat memfasilitasi beberapa transfer yang dilakukan secara paralel. Metode pentransferan filenya masih sama dengan *easy handle*, hanya saja untuk *multi handle*, diperlukan sebuah *handle* tambahan yang dapat menampung seluruh *easy handle* yang akan digunakan. Adapun *handle* tipe ini dapat diinisialisasi dengan memanggil fungsi `curl_multi_init()`, dan untuk mengatur opsi-opsi seputar *multi handle* tersebut, pengguna dapat memanggil fungsi `curl_multi_setopt()`.

Untuk memulai transfer paralel, tentunya perlu diinisialisasi dulu masing-masing *easy handle*-nya. Setelah *handle-handle* tersebut diinisialisasi, *handle* tersebut dapat dimasukkan ke dalam sebuah *multi handle* dengan memanggil fungsi berikut.

```
curl_multi_add_handle(<multi handle>, <easy handle>);
```

Dengan `<easy handle>` merupakan nama dari *easy handle* yang ingin dimasukkan ke dalam *multi handle* tertentu dan `<multi handle>` merupakan nama dari *multi handle*-nya sendiri. Sedangkan, untuk menghapus sebuah *easy handle* dari dari dalam *multi handle*, dapat dipanggil fungsi berikut.

```
curl_multi_remove_handle(<multi handle>, <easy handle>);
```

Setelah seluruh *easy handle* yang ingin dijalankan dimasukkan ke dalam *multi handle*, *multi handle* tersebut dapat dijalankan dengan menggunakan sebuah *loop* transfer. Adapun isi dari loop ini meliputi tiga langkah utama, yaitu:

(a) Inisialisasi variabel `transfers_running`
`transfers_running` merupakan sebuah variabel *integer* yang menjadi penanda bagi pengguna mengenai berapa banyak *handle* yang sedang melakukan proses transfer di suatu waktu. Selama nilai variabel ini bukan 0, artinya ada *easy handle* yang belum selesai melakukan proses transfer berkas.

(b) Menjalankan transfer dalam *multi handle*
Langkah selanjutnya adalah menjalankan transfer dalam *mutli handle*, dengan memanggil fungsi `curl_multi_perform`. Adapun fungsi tersebut harus dipanggil dengan format berikut, yaitu:

```
curl_multi_perform(<multi handle>, <transfers_running>)
```

(c) Menunggu transfer untuk selesai sebelum data diekstraksi
Tentunya sebelum data hasil transfer dapat diekstraksi untuk dipakai, proses transfernya sendiri harus selesai terlebih dahulu—untuk kasus dalam *multi handle* libcurl, seluruh *handle* di dalam *multi handle* harus selesai melakukan transfer terlebih dahulu, atau sampai terjadi *timeout*. Adapun langkah ini dapat diselesaikan entah dengan menggunakan `curl_multi_wait()` atau `curl_multi_poll()`, atau dengan cara manual, yaitu dengan memasukkan deskriptor-deskriptor file serta nilai *timeout* secara manual, dan kemudian menggunakan `select()`, walaupun metode ini tidak dianjurkan karena keterbatasan jumlah deskriptor yang dapat digunakan.[9]

Multi Socket Handle

Ada mode lain dari *multi handle*, yaitu *multi socket handle*. Bedanya dengan *multi handle* biasa adalah *multi socket handle* memperhatikan *socket-socket* yang ada dan memberitahu *handle-handle* jika ada *socket* yang sudah siap untuk digunakan dalam proses *read/write*. Cara operasinya hampir sama dengan *multi handle*—hal utama yang berbeda adalah dengan *multi socket handle*, diperlukan satu buah parameter tambahan, yaitu kumpulan *socket* yang ingin diperhatikan.

[9] https://curl.se/libcurl/c/curl_multi_poll.html

Socket ini, beserta apa aksi yang ingin ditunggu dalam socket tersebut, diperhatikan dengan implementasi sebuah fungsi *callback*, yaitu `socket_callback()`. Handle mana yang ingin digunakan, socket mana yang ingin diperhatikan, serta aksi apa yang ditunggu diatur dalam fungsi ini. Jika ada beberapa *socket* yang ingin diperhatikan, fungsi ini harus dipanggil lagi untuk setiap *socket*-nya. Selain itu, perlu juga dipanggil fungsi `timer_callback()`, yang berfungsi untuk mengatur seberapa lama aplikasi akan menunggu *socket*, sebelum terjadi *timeout*. Kedua *callback* ini dapat diatur ke dalam suatu *multi handle* dengan menggunakan fungsi `curl_multi_setopt()` biasa—untuk *callback socket*, fungsi ini ditandai dengan menggunakan paramter `CURLMOPT_TIMERFUNCTION`, sedangkan untuk *callback timer*, fungsi tersebut ditandai dengan parameter `CURLMOPT_TIMERFUNCTION`.

Adapun seluruh *handle* ini dapat dipanggil dengan memanggil fungsi `curl_multi_socket_action()`, dan cara untuk melihat apakah seluruh transfer sudah selesai atau masih ada transfer yang berlangsung pada suatu waktu sama dengan *multi transfer* biasa.

cJSON[10]

cJSON merupakan sebuah *library* yang berfungsi sebagai *parser* JSON untuk perangkat-perangkat lunak bahasa C. *Library* ini sendiri terdiri atas sebuah file C dan sebuah file header.

Instalasi

cJSON dapat diinstal dengan beberapa cara, yaitu:

- Manual
  Instalasi manual hanya membutuhkan pengembang perangkat lunak untuk menyalin kedua file *library* cJSON ke dalam direktori perangkat lunak tersebut.
- CMake
  Untuk penggunaan cJSON dengan CMake, perlu dibuat sebuah direktori bernama `build`, dan kemudian CMake harus dijalankan di dalam direktori tersebut, dengan mengeksekusi perintah `cmake`. Dengan melakukan ini, Makefile akan dibuat di dalam direktori tersebut, yang nantinya akan dapat di-*compile* dan diinstal dengan perintah `make install`.

  Proses ini akan menginstal file-file *header* ke direktori `/usr/local/include/cjson`, dan file-file *library* ke dalam direktori `usr/local/lib`. Selain itu, file-file untuk `pkg-config` juga akan diinstal untuk memudahkan deteksi instalasi CMake sebelumnya.

  Proses pembangunan cJSON juga memiliki beberapa opsi yang dapat diatur sedemikian rupa sesuai dengan kebutuhan pembuat perangkat lunak. Adapun opsi-opsi yang dapat diatur dapat dilihat di daftar berikut.

  - `-DENABLE_CJSON_TEST`
    Nilai awal: On
    Jika opsi ini dinyalakan (diberi nilai "On"), maka tes-tes cJSON akan dibuat dan dijalankan bersamaan dengan instalasi.
  - `-DENABLE_CJSON_UTILS`
    Nilai awal: Off
    Jika opsi ini dinyalakan (diberi nilai "On"), maka file-file utilitas cJSON akan diinstal bersamaan dengan instalasi.
  - `-DENABLE_TARGET_EXPORT`
    Nilai awal: On
    Mengekspor target-target ekspor cJSON. Opsi ini dapat dimatikan jika terjadi masalah saat instalasi.

[10] https://github.com/DaveGamble/cJSON

  - `-DENABLE_CUSTOM_COMPILER_FLAGS`
    Nilai awal: On
    Mengaktifkan properti-properti untuk *compiler* non-standar (Clang, GCC, MSVC). Opsi ini dapat dimatikan jika terjadi masalah saat instalasi.
  - `-DENABLE_VALGRIND`
    Nilai awal: Off
    Menjalankan tes-tes yang ada menggunakan Valgrind.
  - `-DENABLE_SANITIZERS`
    Nilai awal: Off
    Meng-compile cJSON dengan menyalakan AddressSanitizer dan UndefinedBehaviorSanitizer, jika mungkin.
  - `-DENABLE_SAFE_STACK`
    Nilai awal: Off
    Menyalakan SafeStack. Pada saat skripsi ini dibuat, fitur ini hanya didukung untuk *compiler* Clang.
  - `-DBUILD_SHARED_LIBS`
    Nilai awal: Off
    Membangun *library-library* umum yang tersedia.
  - `-DBUILD_SHARED_AND_STATIC_LIBS`
    Nilai awal: On
    Membangun *library* umum dan *library* statik yang tersedia.
  - `-DCMAKE_INSTALL_PREFIX`
    Nilai awal: -
    Mengatur *prefix* direktori tempat instalasi cJSON.
  - `-DENABLE_LOCALES`
    Nilai awal: On
    Memungkinkan penggunaan metode `localeconv`.
  - `-DCJSON_OVERRIDE_BUILD_SHARED_LIBS`
    Nilai awal: On
    Memungkinkan penimpaan nilai dari opsi `-BUILD_SHARED_LIBS` menggunakan nilai dari opsi `-DCJSON_BUILD_SHARED_LIBS`.
  - `-DENABLE_CJSON_VERSION_SO`
    Nilai awal: On
    Menyalakan versi so dari cJSON.

- Makefile
  Jika CMake tidak tersedia, cJSON juga dapat dibangun dengan menggunakan GNU Make, dengan menggunakan perintah `make all`, dan menginstal *library-library* yang sudah ter-*compile* dengan perintah `make install`. Akan tetapi, perlu diingat bahwa metode instalasi ini sudah tidak lagi diperbarui (*deprecated*), dan dukungannya hanya sebatas pembetulan *bug*.
- vcpkg
  Melalui vcpkg, cJSON dapat diunduh dan diinstal secara langsung. *Port* dari cJSON di vcpkg terus diperbarui oleh tim Microsoft dan kontributor-kontributor dari komunitas publik, jadi versi dari cJSON yang diinstal melalui vcpkg kemungkinan besar akan selalu versi terbarunya.

Penggunaan

Jika cJSON diinstal melalui CMake atau Makefile, cJSON dapat digunakan dengan mengikutkan baris ini dalam kode program:

```
#include <cjson/cJSON.h>
```

Struktur Data

cJSON merepresentasikan sebuah nilai JSON dengan struktur data `cJSON`, yang dapat dilihat di potongan kode 1.

Listing 1: Struktur data cJSON

```
typedef struct cJSON
{
    struct cJSON *next;
    struct cJSON *prev;
    struct cJSON *child;
    int type;
    char *valuestring;
    int valueint;
    double valuedouble;
    char *string;
} cJSON;
```

Dengan variabel-variabel dalam struktur tersebut sebagai berikut:

- `next` dan `prev`
  Variabel ini merupakan penunjuk ke struktur cJSON lainnya yang merupakan nilai JSON selanjutnya (untuk `next`) dan sebelumnya (untuk `prev`).
- `child`
  Variabel ini merupakan penunjuk ke struktur cJSON lainnya yang merupakan elemen JSON di dalam struktur cJSON tersebut.
- `type`
  Variabel ini menandakan tipe dari nilai JSON yang terdapat di dalam struktur cJSON tersebut. Adapun tipe-tipe yang mungkin adalah sebagai berikut.
  - `cJSON_Invalid`
    Merepresentasikan sebuah objek yang tidak valid dan tidak bernilai. Jika seluruh *field* diatur sehingga berisi 0 *byte*, maka variabel `type` akan otomatis berisi tipe ini.
  - `cJSON_True`
    Merepresentasikan nilai booolean *true*.
  - `cJSON_False`
    Merepresentasikan nilai booolean *false*.
  - `cJSON_Number`
    Merepresentasikan nilai numerik apapun. Jika nilainya merupakan nilai *double*, maka nilai tersebut akan disimpan di dalam variabel `valuedouble`. Sedangkan jika nilainya merupakan nilai *integer*, maka nilai tersebut akan disimpan di dalam variabel `valueint`.
  - `cJSON_String`
    Merepresentasikan nilai *string* apapun. Nilainya disimpan sebagai *string* yang dipisah dengan *null terminator* (`'\0'` atau `\u0000`).
  - `cJSON_Array`
    Merepresentasikan nilai *array*. *Array* dalam cJSON diimplementasikan dengan menunjukkan isi dari variabel `child` tadi ke sebuah *linked list* dari objek-objek cJSON. Jika objek cJSON

ini merupakan elemen dalam sebuah *array*, maka isi dari variabel `prev` dan `next` merupakan salah satu dari elemen-elemen lain dalam *array* yang menjadi elemen sebelum dan sesudah elemen ini.

  - `cJSON_Object`
    Merepresentasikan nilai objek general. Implementasinya sama dengan *array*, hanya saja untuk objek general, kuncinya diletakkan di dalam variabel `string`.
  - `cJSON_Raw`
    Merepresentasikan objek JSON apapun yang disimpan dalam bentuk *array* yang diterminasi/dipisah oleh *null terminator*.
  - `cJSON_NULL`
    Merepresentasikan sebuah nilai *null*.

- `valuestring`/`valuestring`/`valuestring`/`string`
  Merupakan variabel-variabel yang menyimpan nilai dari struktur cJSON. Variabel apa yang diisi dan apa isinya tergantung dari tipe data yang disimpan.

Fungsi-Fungsi Dasar

- Tipe-tipe data dasar
  Untuk setiap tipe data cJSON, ada sebuah fungsi `cJSON_Create...`. Semua fungsi tersebut akan mengalokasikan sebuah struktur cJSON yang nantinya dapat dihapus dengan `cJSON_Delete`. Perlu diingat juga bahwa seluruh struktur yang dibuat harus dihapus agar tidak terjadi kebocoran memori. Adapun fungsi-fungsi `cJSON_Create` yang dapat digunakan untuk tipe-tipe data yang tersedia adalah sebagai berikut.
  - *null* dibuat dengan `cJSON_CreateNull`.
  - *boolean* dibuat dengan `cJSON_CreateBool`, atau jika ingin membuat data *boolean* langsung dengan nilainya, dapat menggunakan `cJSON_CreateTrue` atau `cJSON_CreateFalse`.
  - Bilangan apapun dibuat dengan `cJSON_CreateNumber`. Penggunaan fungsi ini akan langsung mengisi nilai variabel `valueint` dan `valuedouble`.
  - *string* apapun dibuat dengan `cJSON_CreateString` (membuat sebuah string langsung sebagai nilai data JSON), atau `cJSON_CreateStringReference` (mereferensikan *string* yang sudah ada sebagai nilai data JSON).

- *Array*
  Sebuah *array* kosong dapat dibuat dengan menggunakan fungsi `cJSON_CreateArray`. Selain fungsi tersebut, pembuatan *array* juga dapat dilakukan dengan memanggil sebuah fungsi alternatif, yaitu `cJSON_CreateArrayReference`. Bedanya adalah dengan fungsi alternatif ini, *array* yang dibuat tidak secara langsung "mengandung" nilai-nilainya, jadi ketika *array* tersebut dihapus dengan `cJSON_Delete`, nilai-nilai di dalamnya tidak terhapus juga. Hal yang sama juga dapat dilakukan dengan objek-objek di dalam *array* tersebut, dimana objek ini dapat ditambahkan dengan fungsi `cJSON_AddItemToArray`, yang akan langsung menambahkan objek tersebut ke dalam *array*-nya, atau dengan menggunakan `cJSON_AddItemReferenceToArray`, yang hanya akan menambahkan referensi dari objek yang ingin ditambahkan ke dalam *array*. Jika ada sebuah objek yang ingin dihapus dari *array* tertentu, perangkat lunak dapat memanggil fungsi `cJSON_DetachItemFromArray`. Fungsi ini akan mengembalikan objek yang dihapus dari *array* tadi, jadi objek ini harus diberikan ke sebuah penunjuk lainnya (dimasukkan ke dalam variabel lain) agar tidak terjadi kebocoran memori. Selain fungsi tersebut, fungsi `cJSON_DeleteItemFromArray` juga dapat digunakan—bedanya adalah objek yang dihapus dari *array* tadi, jika dihapus menggunakan fungsi ini, akan langsung dihapus, seakan-akan fungsi `cJSON_Delete` dipanggil untuk objek

tersebut. Untuk menimpa/mengganti nilai suatu objek di dalam *array*, fungsi `cJSON_ReplaceItemInArray` dapat dipanggil dengan indeks dari objek yang ingin diganti sebagai parameternya. Selain fungsi tersebut, fungsi `cJSON_ReplaceItemViaPointer` juga dapat digunakan—fungsi ini bekerja dengan memutuskan (*detach*) objek lama yang ingin diganti, menghapus objek tersebut, dan menyisipkan objek yang baru ke tempat objek lama yang sudah dihapus tadi. Terakhir, untuk mendapatkan objek tertentu dalam *array* berdasarkan indeksnya, perangkat lunak dapat memanggil fungsi `cJSON_GetArrayItem`. Ukuran dari *array*-nya sendiri juga dapat dilihat dengan fungsi `cJSON_GetArraySize`.

- Objek
  Sama seperti *array*, ada dua jenis fungsi pembuatan objek. Yang pertama adalah `cJSON_CreateObject` untuk membuat objek biasa, dan `cJSON_CreateObjectReference` untuk membuat objek yang nilai-nilainya terdiri atas kumpulan referensi ke variabel-variabel lain. Untuk menambahkan sebuah objek ke dalam suatu objek lainnya, pengguna dapat memanggil fungsi `cJSON_AddItemToObject`. Jika objek ingin ditambahkan ke suatu objek lain yang memiliki sebuah variabel konstan sebagai nama, atau sebuah referensi, prosesnya harus menggunakan fungsi `cJSON_AddItemToObjectCS`. Fungsi `cJSON_DetachItemFromObjectCaseSensitive` dapat dipanggil ketika ada sebuah objek ingin dibuang dari objek lain. Sama seperti fungsi *detach* di *array* tadi, fungsi ini akan mengembalikan objek yang dibuang tadi, dan objek tersebut harus dimasukkan ke variabel lain untuk menghindari kebocoran memori. Selain itu, ada juga fungsi `cJSON_DeleteItemFromObjectCaseSensitive`, yang bekerja dengan cara yang sama seperti fungsi penghapusan objek untuk *array*. Untuk pengeditan/penggantian nilai objek, lagi-lagi cara kerjanya sama dengan *array*. Untuk penggantian nilai objek berdasarkan kuncinya, fungsi `cJSON_ReplaceItemInObjectCaseSensitive` dapat digunakan, sedangkan untuk penggantian objek langsung dengan penunjuk ke elemen lainnya, fungsi `cJSON_ReplaceItemViaPointer` dapat digunakan. Terakhir, untuk mengakses sebuah benda di dalam objek, pengguna dapat memanggil fungsi `cJSON_GetObjectItemCaseSensitive`, dan untuk mengetahui ukuran dari objek tersebut, fungsi yang dapat digunakan sama dengan fungsi yang dapat digunakan untuk *array*, yaitu `cJSON_GetArraySize`.
- *Parsing*
  Objek JSON yang dibatasi/diterminasi dengan *null terminator* dapat di-*parse* dengan menggunakan fungsi berikut.

  ```
  cJSON_Parse(<JSON>)
  ```

  Sedangkan, jika objek JSON tersebut tidak (atau belum tentu) dibatasi oleh *null terminator*, objek tersebut dapat di-*parse* dengan fungsi berikut.

  ```
  cJSON_Parse(<JSON>, <ukuran JSON yang ingin di-parse>)
  ```

  Kedua fungsi ini akan membuat sebuah struktur hierarkial dari objek-objek cJSON yang merepresentasikan keseluruhan dari objek JSON tersebut. Setelah objek ini selesai digunakan, penggunanya harus mendealokasikan objek tersebut dengan fungsi `cJSON_Delete`.

CMake [4]

CMake merupakan sebuah perkakas generator *build system* yang memungkinkan pengembang perangkat lunak untuk menentukan parameter-parameter pembangunan perangkat di sebuah file yang berupa teks biasa. File ini kemudian dipakai oleh CMake untuk membuat perkakas-perkakas pembangunan perangkat lunak yang dapat dibaca oleh IDE-IDE tertentu, seperti Microsoft Visual Studio, Apple Xcode, Linux, dan sebagainya. Selain itu, CMake juga menangani aspek-aspek rumit dari pembangunan perangkat lunak, seperti pembangunan perangkat lunak *cross-platform*, introspeksi sistem, dan juga pembangunan perangkat lunak yang dapat disesuaikan berdasarkan penggunanya.

Untuk proyek-proyek yang dibangun di satu platform, CMake memiliki fungsi sebagai berikut.

- Memberikan kemampuan untuk melakukan pencarian seluruh perangkat lunak, file-file *library*, dan file-file *header* yang dibutuhkan untuk proses pembangunan perangkat lunak. Pencarian ini juga dapat dilakukan sampai ke dalam *registry* sistem.
- Memungkinkan pembangunan perangkat lunak diluar folder tempat *source code* perangkat lunak tersebut sendiri.
- Memberikan kemampuan untuk membuat perintah-perintah khusus untuk file-file yang dibuat secara otomatis, seperti `moc()` dari Qt, atau generator pembungkus dari SWIG. Perintah-perintah ini dapat mengatur pembuatan file *source* baru yang nantinya akan diintegrasikan langsung ke dalam perangkat lunak akhirnya.
- Memberikan kemampuan untuk memilihkan file-file opsional dari *library* yang digunakan.
- Memungkinkan pengaturan pembuatan proyek dari sebuah file `.txt` sederhana.
- Kemampuan untuk dengan mudah beralih dari *static build* dan *shared build*.
- Membuat ketentuan keperluan (*dependency*) secara otomatis.

Sedangkan, untuk perangkat-perangkat lunak yang dibuat *cross-platform*, CMake memberikan fungsi-fungsi tambahan sebagai berikut.

- Memberikan kemampuan mengetes urutan *machine byte* dan karakteristik-karakteristik lainnya yang spesifik untuk suatu sistem operasi.
- File konfigurasi yang dibuat dapat berlaku untuk seluruh *platform*.
- Memberikan dukungan untuk membangun *shared build* untuk seluruh *platform* yang mendukungnya.
- Memberikan kemampuan untuk mengatur opsi-opsi yang spesifik untuk suatu sistem operasi, seperti misalnya lokasi file-file data utama.

Penggunaan Dasar

CMake mengambil satu (atau lebih) file-file CMakeLists sebagai masukan, dan sebagai keluaran akan menghasilkan file-file proyek atau Makefile yang dapat digunakan dengan dan oleh perkakas-perkakas pembangunan perangkat lunak lain yang ada.

Proses CMake umumnya terdiri atas langkah-langkah berikut.

(a) Proyek yang ingin dibangun didefinisikan di salah satu file CMakeLists.
File-file CMakeLists (yang berupa file-file `.txt`) merupakan file-file *plain text* yang berisi deskripsi proyek dalam bahasa CMake. Tertera di kode 2 merupakan file CMakeLists yang dapat digunakan untuk membangun sebuah program *"Hello World"* sederhana dalam bahasa C, sebagai gambaran mengenai hal-hal apa saja yang minimal harus ada di dalam file CMakeLists.

Listing 2: Kode utama operasional CMakeLists

```
cmake_minimum_required(VERSION 3.20)
project(Hello)
add_executable(Hello Hello.c)
```

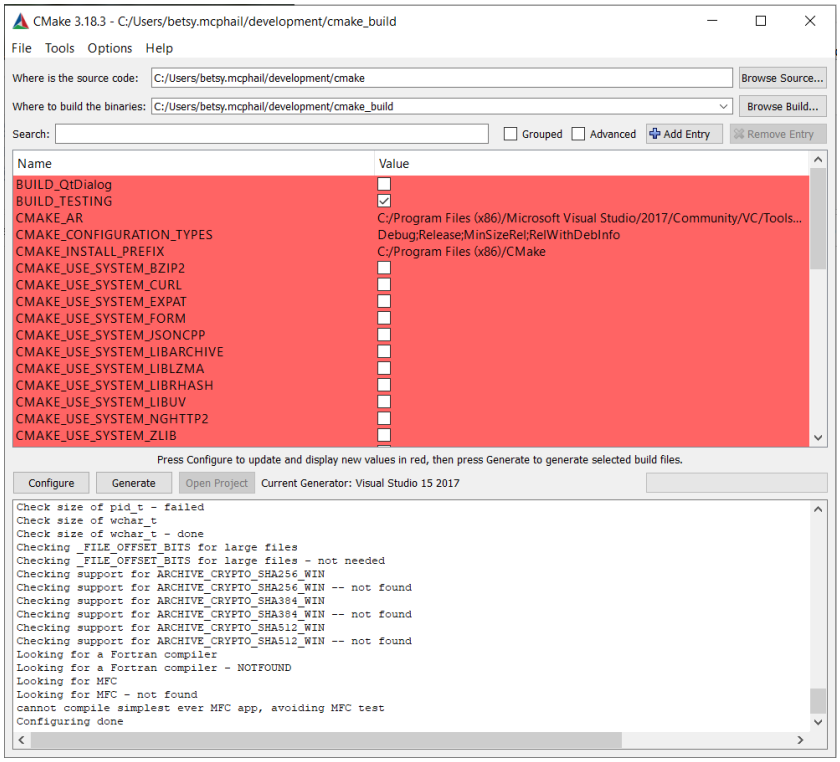
Penjelasan dari baris-baris berikut adalah sebagai berikut.

- Baris pertama harus selalu merupakan `cmake_minimum_required`. Hal ini mengharuskan proyek untuk menggunakan versi dari CMake yang dispesifikasi, dan juga memungkinkan *backwards compatibility*.

- Baris selanjutnya merupakan perintah `project`. Perintah ini mengatur nama proyek, dan juga bisa digunakan untuk mengatur aturan-aturan lainnya, seperti versi, atau bahasa dari proyek. Untuk setiap direktori yang memiliki file CMakeLists, CMake akan membuat sebuah Makefile atau file proyek untuk IDE. Proyek ini, nantinya, akan mengikutkan seluruh direktori yang memiliki file CMakeLists ini serta seluruh subdirektori yang diikutkan dalam perintah `add_subdirectory`.
- Terakhir, perintah `add_executable` merupakan sebuah perintah yang akan menambahkan sebuah file *executable* untuk menjalankan proyek yang sudah dibangun tersebut.

(b) CMake membuat dan mengkonfigurasi proyek tersebut.
Setelah file-file CMakeLists selesai dibuat, CMake akan memproses file-file tersebut dan membuat entri-entri dalam sebuah file *cache*. Pengembang perangkat lunak dapat mengedit file CMakeLists atau mengatur isi dari file *cache* tadi dengan GUI CMake, ccmake, atau untuk proyek-proyek skala kecil, langsung dari *command line*.

- GUI CMake
  CMake memiliki perangkat lunak GUI berbasis Qt yang bisa digunakan di sebagian besar sistem operasi, seperti UNIX, Mac OS X, dan Windows. Perangkat lunak ini, `cmake-gui`, sudah terinstal bersama dengan CMake, tetapi membutuhkan instalasi Qt untuk dijalankan. Adapun tampilan dari perangkat lunak ini dapat dilihat di gambar 8.



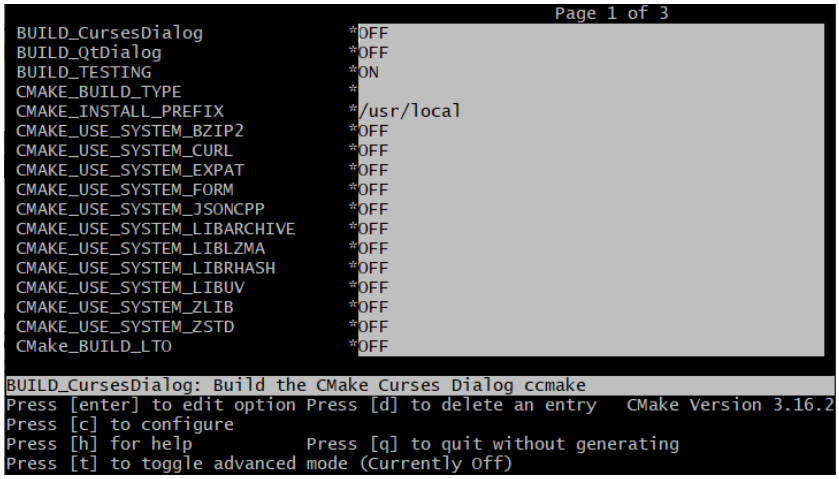
Gambar 8: Tampilan aplikasi cmake-gui.[11]

Dua *field* paling atas adalah direktori dari *source code* dan direktori tempat file-file *binary* nantinya akan diletakkan setelah dibuat. Kedua *field* ini harus diisi secara manual, walaupun jika direktori *binary* ini sudah dikonfigurasi langsung melalui CMake sebelumnya, *field* direktori kedua akan secara otomatis terisi.

- ccmake
  Di mayoritas sistem operasi berbasis UNIX, jika *library curses* didukung, maka CMake me-

[11] https://cmake.org/cmake/help/book/mastering-cmake/chapter/Getting%20Started.html

miliki sebuah perangkat lunak lain yang dapat digunakan, yaitu `ccmake`. Untuk menjalankan `ccmake`, pengguna dapat menjalankannya melalui *command line*, dan direktori tempat `ccmake` dijalankan ini harus merupakan direktori tempat file-file *binary* nantinya ingin disimpan. Ketika aplikasi ini dijalankan, akan keluar tampilan seperti di gambar 9. Adapun instruksi singkat penggunaan dari aplikasi ini dapat dilihat di bagian bawah dari tampilan tersebut.



Gambar 9: Tampilan aplikasi ccmake.[12]

- Langsung dari *command line*
  CMake juga dapat dijalankan melalui *command line*. Untuk menjalankan CMake dari *command line*, direktori tempat terminal berada lagi-lagi harus diatur ke direktori tempat file-file *binary* akan disimpan. Kemudian jalankan perintah `cmake` dengan opsi `-D`, diikuti dengan direktori tempat *source code* dari perangkat lunak yang ingin dibangun berada. Walaupun begitu, perlu diingat bahwa metode ini direkomendasikan untuk digunakan hanya untuk proyek-proyek yang memiliki sedikit, atau bahkan tidak ada opsi sama sekali.

(c) Pengguna membangun proyek tersebut dengan perkakas pembangunan masing-masing.
CMake dapat memberikan beberapa opsi dalam pembangunan perangkat lunak yang dapat diatur oleh penggunanya masing-masing. Adapun dua opsi utama dari seluruh opsi-opsi tersebut adalah sebagai berikut.

- Menspesifikasi *compiler* yang akan digunakan
  Di beberapa sistem, bisa jadi terdapat lebih dari satu *compiler*, atau *compiler* yang ada tidak berada di tempat *compiler* tersebut biasa diinstal. Di kasus-kasus ini, CMake perlu diberitahu secara manual dimana letak *compiler* yang ingin digunakan. Ada tiga cara untuk melakukan ini, yaitu dengan:
  - menspesifikasikan *compiler* yang ingin dipakai ke generator,
  - mengatur variabel *environment*, atau
  - membuat entri *cache*.

  Jika pengaturan *compiler* sudah selesai dan `cmake` sudah dijalankan setidaknya sekali, jika pengguna ingin mengganti *compiler*, pengguna harus memulai ulang dengan folder file *binary* yang kosong.
- Mengatur konfigurasi pembangunan perangkat
  Konfigurasi pembangunan perangkat memungkinkan sebuah proyek untuk dibangun dalam beberapa cara dengan tujuan *debugging*, optimisasi, atau tujuan-tujuan sejenis lainnya. Secara *default*, CMake mendukung mode-mode berikut:

[12] https://cmake.org/cmake/help/book/mastering-cmake/chapter/Getting%20Started.html

- `Debug`—Opsi-opsi *debug* dasar dinyalakan.
- `Release`—Fungsi optimisasi dasar dinayalakan.
- `MinSizeRel`—Ukuran kode akhir diusahakan sekecil mungkin.
- `RelWithDebinfo`—Membangun *build* optimal dan mengikutkan informasi-informasi untuk *debugging*.

Dengan generator-generator berbasis Makefile, hanya sebuah mode konfigurasi yang bisa aktif ketika CMake sedang dijalankan, dan mode tersebut diatur dengan variabel `CMAKE_BUILD_TYPE`. Jika variabel ini dikosongkan (atau tidak dimasukkan ke dalam kode), maka mode yang dipakai adalah mode *default*. Di lain kasus, jika pengembang ingin membangun perangkat dalam dua mode yang berbeda, perintah untuk menjalankan CMake (dengan variabel mode konfigurasi masing-masing) harus dijalankan untuk setiap modenya. Misal, jika pengguna ingin membangun perangkat dengan mode `Debug` dan `Release` sekaligus, maka perintah untuk menjalankan CMake harus ditulis seperti dapat dilihat di potongan kode 3.

Listing 3: Kode utama operasional CMakeLists

```
ccmake ../<direktori> -DCMAKE_BUILD_TYPE=Debug
ccmake ../<direktori> -DCMAKE_BUILD_TYPE=Release
```

Setelah CMake dijalankan, proyek tersebut siap untuk dibangun. Jika generator yang dipilih merupakan generator berbasis Makefiles, maka proyek tersebut dapat dibangun dengan mengganti *working direktory* ke lokasi file-file *binary*, dan kemudian menjalankan perintah `make`. Jika generator yang dipilih merupakan sebuah IDE, maka proyek tersebut dapat dibangun secara biasa melalui IDE tersebut.

7. **Membangun perangkat lunak berdasarkan rancangan yang sudah dibuat, dengan megimplementasikan seluruh modul dan *library* yang telah ditentukan di tahap sebelumnya dalam bahasa C.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Poin ini akan dikerjakan di Skripsi 2.

8. **Melakukan pengujian fungsional dan perbaikan *bug*.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Poin ini akan dikerjakan di Skripsi 2.

9. **Menulis dokumentasi perangkat lunak.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Poin ini akan dikerjakan di Skripsi 2.

10. **Menulis dokumen skripsi.**
    Status : Ada sejak rencana kerja skripsi.
    Hasil : Dokumen skripsi akan ditulis hingga bab 3 pada Skripsi 1. Pengisian sisa dari dokumen skripsi, serta penyempurnaan bab 1 sampai bab 3, akan dilakukan pada Skripsi 2.

7 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Skripsi 1 ini adalah sebagai berikut:

1.
2.
3.

Daftar Referensi

[1] Marsh, N. (2010) *Introduction to the Command Line: The Fat-Free Guide to Unix and Linux Commands*, 2nd edition. CreateSpace, South Carolina.

[2] The GNU C Library (2022) cd. https://www.gnu.org/software/libc/manual/html_node/Getopt.html. versi 2.3.5.

[3] Stenberg, D. (2022) *Everything curl*. https://everything.curl.dev/.

[4] Kitware (2022) *Mastering CMake*. https://cmake.org/cmake/help/book/mastering-cmake/.

Bandung, 20/03/2022

Alfred Aprianto Liaunardi

Menyetujui,

Nama: Pascal Alfadian Nugroho, M.Comp.
Pembimbing Tunggal