1장
Hello, my name is 광연.

I am a leader of the Hanjp project, and I’m currently working for the mentoring program called Contribution Academy nowadays.

Today, I’m introducing what is the Hanjp,
a history of the project,
why are we doing this project,
and a little bit about technicals.

2장

So, what is the Hanjp?

The Hanjp is a Japanese input method, which enables typing Japanese string as the Korean Hangul.

Hangul is the Korean alphabet.

Koreans, we write sentences as Hangul.

The theory of Hangul representation for other languages is not a new idea.

This idea is rather natural to Koreans.

Especially, the Japanese Kana string representation in Hangul is pretty intuitive.

Some of the apps, especially in web services, support Hangul to Kana string conversion.

However, surprisingly, I noticed there is no Input Method that enables us to write Japanese as Hangul for many apps.

I have valued this theory as it could be very useful in Japanese IME, and it seemed fun.

The typing experience in Hanjp is similar to Hangul typing, but it actually produces Japanese string.

3장
I have an example.
If we want to type 힛시 to some app in Hiragana, we type 힛시 just like we are typing in Hangul.
However in the typing process, the engine converts Hangul pre-edit string to kana pre-edit string 3 times.
Then finally, it commits the Hiragana string to the app.

4장
I prepared the demo video. Let me show you.

5장
So, that’s the Hanjp.
I believe that the Hanjp project is simple, convenient, and fun.
However, I also reckon the fundamental idea in Hanjp couldn’t, in turn, make a huge improvement for Japanese typing.
It is just an option, finally.
However, I started this project because I wanted to understand Linux application development, and I also wanted to be involved in open source.
Then, I thought it is a fit project to study Linux App development because the fundamental idea in Hanjp is not super complicated.
Then, what is the value of Hanjp, finally?
I valued this project as an entry point for Korean open source newcomers.
Because I had many difficulties in involving open source contributions, and still I have, so I’ve wanted to help other open source newcomers.
And I believe Hanjp could be a good entry point for open source newcomers because Hanjp is based on some Linux commonly used technologies, still it is not a super complex to understand.

6장
For now, let me talk about our history.
I gathered project members in 2017.
Then, I could get 4 members.
Youngbin was one of the members, and he was the leader of Ubuntu Loco Team.
Talking with him, we thought it can improve linux usability, so we decided to host a project on Ubuntu Loco Team.
Then, we started the project in early 2018.

7장
Before we got started, I thought that we could achieve our goal easily, although we didn’t know much about Linux IME background knowledge.
Because the fundamental idea in Hanjp is quite simple, and its implementation for conversion logic is not also super complicated.
However, it was rather a journey.
There’s a lot of knowledge under Linux IME such as Unicode encoding, build systems, gettext, and GLib.
And the knowledge is required for engine implementation.
Furthermore, there were a lack of documents and easily understandable ones were even few.
So, I had many difficulties in managing the team.
As time went by, the team members left one by one.
Then, even Youngbin, the last member who worked with me, has left for his military service.

8장
However, since I have responsibility for this project, I restarted the project alonely.
I read the ibus-hangul engine and libhangul code repeatedly.
In this process, I have studied many kinds of subjects such as Autotoolset, CMake, iBus, and GLib.
Then, I finally made an executable version of Hanjp.
The demo video we previously watched together was this version.

9장

I appreciate the support given by some people.
First, thank you for sharing the working room, Byongseung,.
Youngbin introduced him to me, and he candidly shared the working room for Hanjp.
Thank you for your support, Byougseung.
Then, thank you for organizing support, Youngbin .
He organized many kinds of histories and he studied a lot of linux development subjects with me.
The executable version of Hanjp based on this kind of his efforts.
Thank you Youngbin.
They patiently endured a lot of my mistakes, for sure.
조병승님, 한영빈님 정말로 감사합니다.
And I also appreciate other contributions made by a lot of people.
Anyway, making an executable version is a very hopeful achievement.
However, we are hardly refining the engine nowadays because this version was insufficient for stable release.
In that state, the project was not a joinable to other people.

10장
As I said, we are refining the project in an academic program called Contribution Academy.
The Contribution Academy is a mentoring program for Korean open source newcomers.
For two months, mentees study about how they can contribute to open source in one specific project.
Even though our project is not perfectly ready for contributors, I joined this program as a mentor because I thought that now I can share the knowledge that is useful for Linux IME development.

11장
So, it’s been about 3 weeks.
In the first, I shared the basics such as WSL, Fundamentals of OOP, and a little of CMake, before we dive into implementation.
Now, our subjects are GObject and CMake.
Studying CMake and GObject, we are redesigning libhanjp the subroutines for the Hanjp engine.
I had sketched the basic design on GObject and I left almost all member functions as black for mentees.
Our ambitious goal in this program is a stable release for Hanjp.
It is not going super well, but I think this flow is still hopeful.
I think we can still release the stable.

12장
We have an online meeting periodically, and this is the picture taken at our online meeting.

13장
Anyway, let’s talk about some technical things.
We are commonly using GLIb and GObject in this project.
One of our main goals is to implement the ibus Hanjp engine.
In the engine, it depends on libkkc for the Kanji conversion.
Then, by implementing the Hanjp engine, we also produce commonly usable Hangul Kana string pre-editing logics.
We call it as libhanjp.
The libhanjp can be portable to many targets including windows, mac, even microcontrollers, thanks to the nature of its own code and dependencies.
However, currently our CMake build directing code can only build on Linux systems, even though CMake supports cross-platform.
Making it possible to build on other environments is also one of our main goals.

14장
Let’s go a little deeper into libhanjp.
The main entry point for libhanjp is InputContext.
It binds ASCII code to Hangul jaso in Keyboard object, and the taken jaso into Automata, and automata works for pre-editing including Hangul to Kana conversion.Let’s go deep into libhanjp.
In the Hanjp project, I learned pluggable design is important.
And the pluggable design is rated on polymorphism.
A pluggable design makes it possible to switch sub-routines in runtime.
For example, we can switch automata logic in runtime by the switching of automata instance
Because the interface is the same in any automata, we can do it easily.
Furthermore, we can easily change dependencies that rely for implementations, thanks to the interface.
Then, we are supporting dynamic custom automata registration.

15장

I strongly believe that automata under the Hanjp shouldn’t be determinative.

Because Hanjp is more close to experimental, a better or different types of automata ideas can alway be suggested by the users.

And their implementations should be allowed by the pluggable customs for most swift usability experiments, and for fun.

16장
Originally, we decided to write the code in C rather than C++.
Since iBus API is written in C, I thought there's a risk for working on C++.
However I've noticed that libhanjp should be designed under OOP.
In C style, it was much less readable.
Plus, I found that C can cause code fragmentation because of its language nature.
The C standard is really tiny.
It doesn't even support fundamental data structures such as linked list, dictionary, and so on.
Naturally, there will be many implementations for these fundamentals.
Finally, it could cause painful code re-inventions and integrations.
GLib can be a one of the solutions, but I thought that GLib was very difficult to use.
Anyway, C++ provides a much bigger standard.
So, I think C++ code is much more readable than C.
Plus, the calling of C subroutines in C++ main routine is easy.
So, I thought migrating the code base to C++ wouldn't cause problems.

17장
However, we are returning to C.
There are several reasons.
First, I couldn’t find a way to design the engine with GObject in C++.
For the ibus engine, we should design the engine on GObject, but it couldn’t on C++.
Then, I found out the advantages of working on GObject.
First, It supports dynamic type registration.
As previously I said, we need custom automata registration, and GObject seemed fit for requirements rather than other techniques such as lua language.
Second, we can support other language bindings effortlessly with GObject.
GObject can automatically generate glue code for other programming languages.
Lastly, GObject is portable to many platforms more than C++.
Because the C compiler is much more than C++ especially when it comes to small devices, and GObject written in C.
Thus, we can easily produce the library for many kinds of target devices, even for microcontrollers.

18장
So, that was my introduction for Hanjp.
Lastly, let me talk about the future of Hanjp that I think.
I believe Hanjp can be the playground for Korean university students.
Because Hanjp isn’t super complicated, I believe Hanjp code would remain as understandable.
Then, continuously teaching the backgrounds to Linux newcomers, I believe the Hanjp project can produce more people who understand GLib and Linux IME, and it’s good for Linux related communities.
Then, making good Japanese IME, libhanjp can be the feature rich framework that is useful to some other apps, and I believe it could be used in some industries.

19장
Yap, that was my presentation.
And, thank you for listening