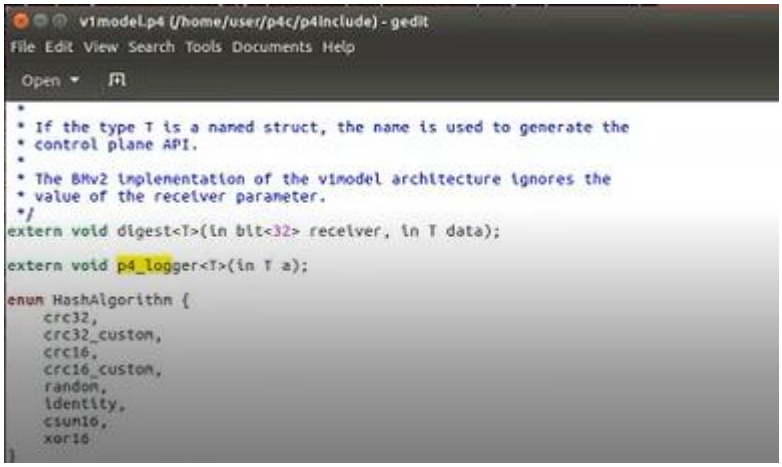
0615 在 p4 程式裡多一個函式：p4_logger

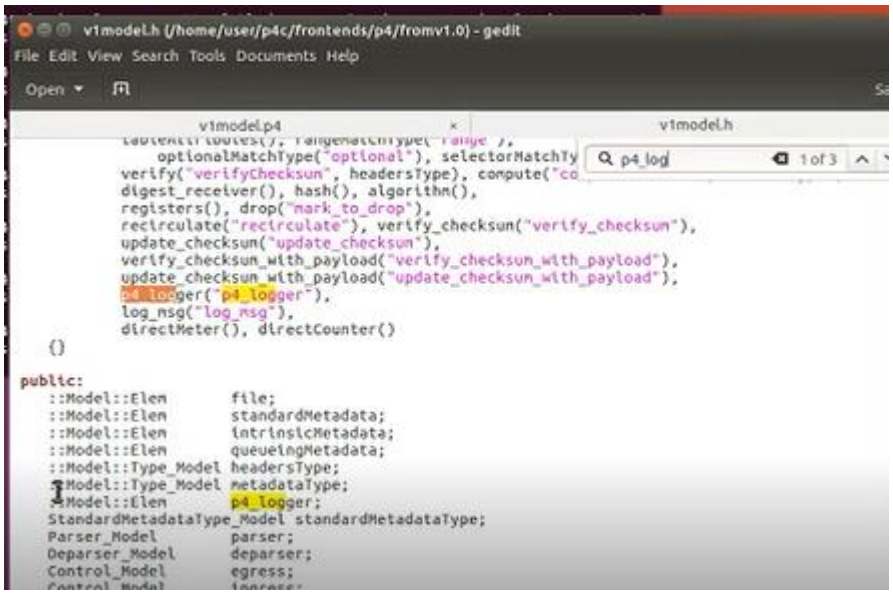
在開始程式的時候，想把那些東西印出來就可以印出來

安裝步驟：

1. 先到 https://github.com/cslev/p4extern 網站
2. 打開終端機，切到 p4-test，執行 gedit &
3. Open -> other documents -> user -> p4c -> p4include -> v1model.p4
4. 把 extern void p4_logger<T>(in T a);加上

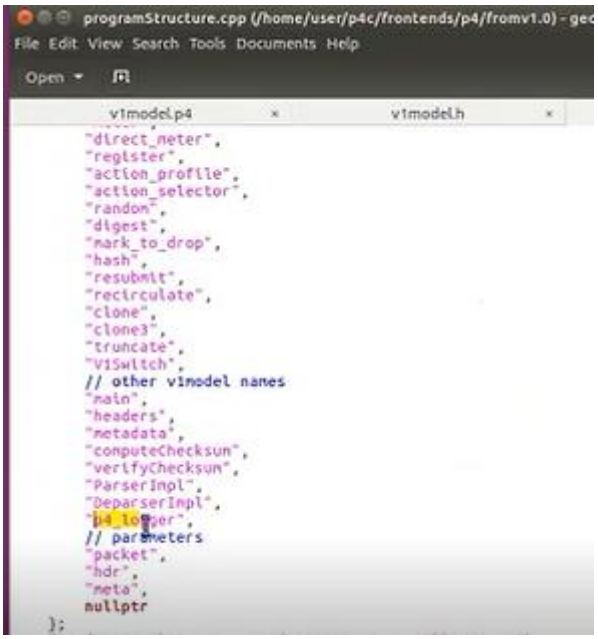


5. 加上以後，繼續加下一個。一樣 Open -> other documents
6. user -> p4c -> frontends -> p4 -> fromv1.0 -> v1model.h
7. 加入 `p4_logger("p4_logger"), /` ::Model::Elem p4_logger;

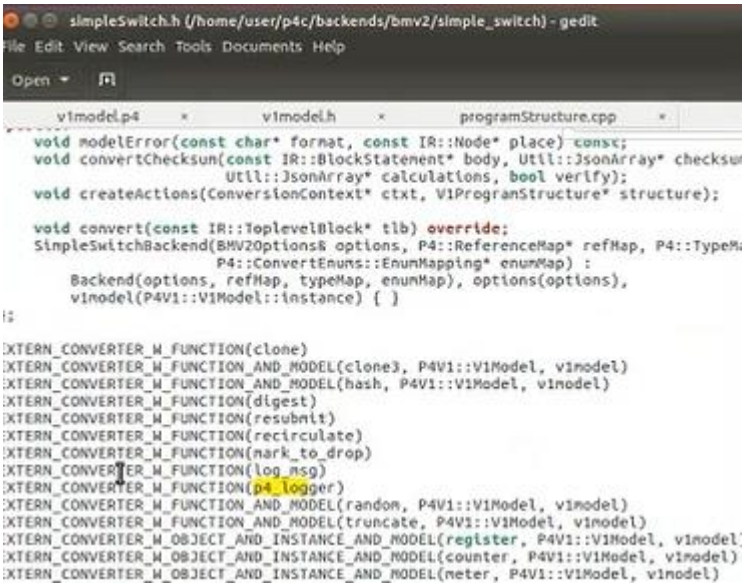


8. 繼續加下一個。Open -> other documents
9. user -> p4c -> frontends -> p4 -> fromv1.0 -> programStructure.cpp

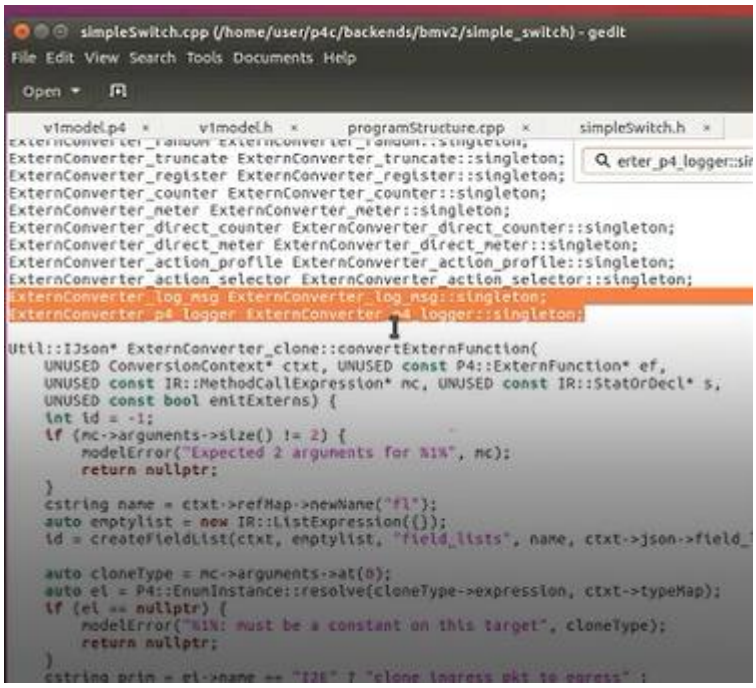
10. 加入　"p4_logger",



11. 繼續加下一個。Open -> other documents
12. user -> p4c -> backends -> bmv2 -> simple_switch -> simpleSwitch.h
13. 加入 EXTERN_CONVERTER_W_FUNCTION(p4_logger)



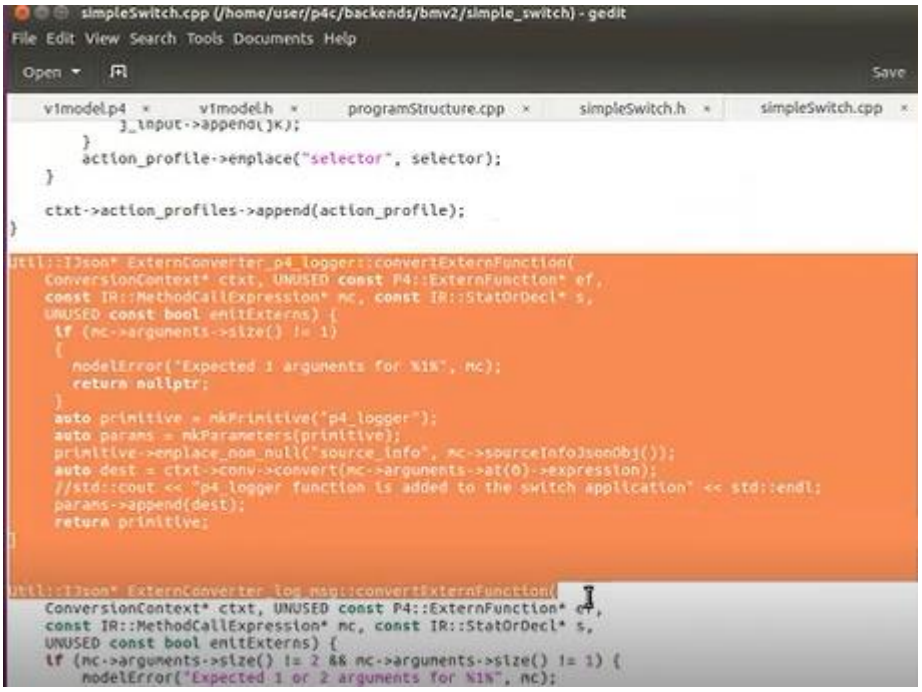
14. 繼續加下一個。Open -> other documents
15. user -> p4c -> backends -> bmv2 -> simple_switch -> simpleSwitch.cpp
    這個地方照抄網站上會錯！要照老師的改！
16. 加入 ExternConverter_p4_logger ExternConverter_p4_logger::singleton;



17. 同檔案還有另一個地方要加，加入整段：

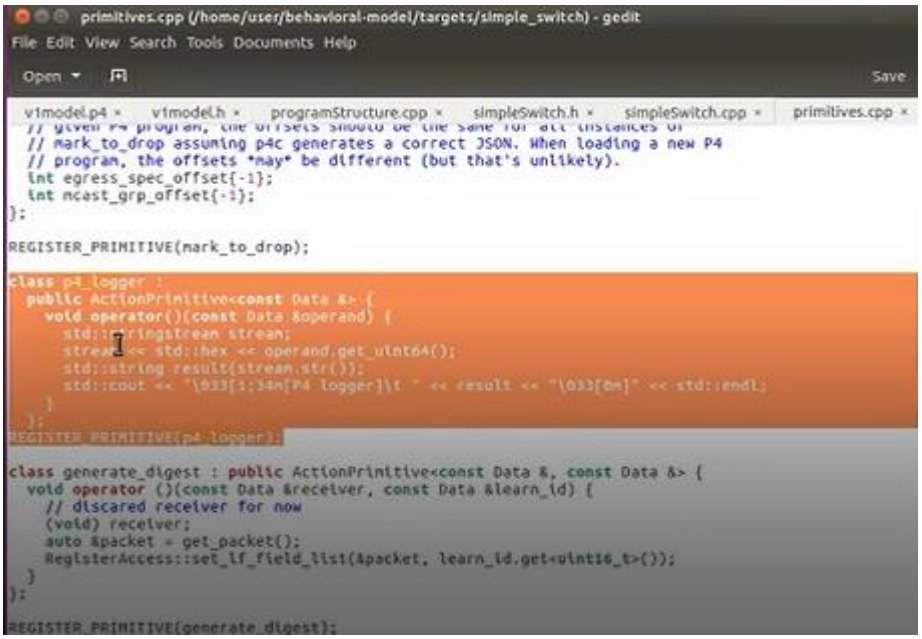
```
Util::IJson* ExternConverter_p4_logger::convertExternFunction(
        ConversionContext* ctxt, UNUSED const P4::ExternFunction* ef,
        const IR::MethodCallExpression* mc, const IR::StatOrDecl* s,
        UNUSED const bool emitExterns) {
          if (mc->arguments->size() != 1)
          {
             modelError("Expected 1 arguments for %1%", mc);
             return nullptr;
          }
          auto primitive = mkPrimitive("p4_logger");
          auto params = mkParameters(primitive);
          primitive->emplace_non_null("source_info", mc->sourceInfoJsonObj());
          auto dest = ctxt->conv->convert(mc->arguments->at(0)->expression);
          //std::cout << "p4_logger function is added to the switch application" <<
  std::endl;
          params->append(dest);
          return primitive;
}
```

整段插在 Util::IJson* ExternConverter_log_msg::convertExternFunction(上面



18. 最後一個加入。Open -> other documents
19. user -> behavior-model -> targets -> simple_switch -> primitives.cpp
20. 加入整段

```
class p4_logger :
    public ActionPrimitive<const Data &> {
        void operator()(const Data &operand) {
            std::stringstream stream;
            stream << std::hex << operand.get_uint64();
            std::string result(stream.str());
            std::cout << "\033[1;34m[P4 logger]\t " << result << "\033[0m]" <<
std::endl;
        }
    };
REGISTER_PRIMITIVE(p4_logger);
```

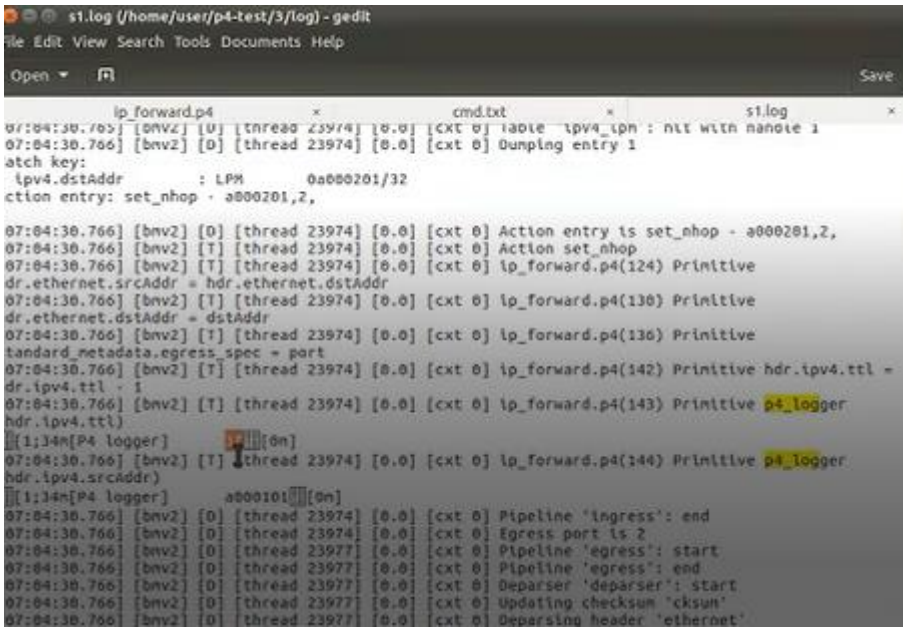


21. 做完以後，程式碼要重新編譯。把改好的程式碼 save 並關掉
22. 切到 user/p4c/build 資料夾
23. 執行 make –j4
24. 執行 make install
25. 跑完後切到 user/ behavior-model /targets/simple_switch 資料夾
26. 執行 make –j4
27. 執行 make install

沒有出錯就可以開始用了！

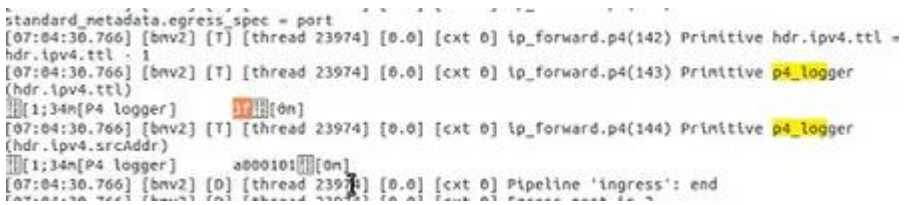
執行步驟：

1. 打開終端機，切到 p4-test/3 資料夾
2. gedit ip_forward.p4 &
3. p4run
4. mininet 執行 h1 ping –c 3 h2，ping 完結束 exit
5. gedit 那裏打開 Open -> other documents -> log -> s1.log
6. 就會看到值是多少，例如說出現 3f，也就是說現在的 ttl 是 3f



7. 打開小算盤，左上角三條線選擇程式設計人員，選擇 16 進位(HEX)
8. 輸入 3f，十進位就是 77

把來源 ip 印出來



a0000101 這個封包就是 10.0.1.1

在程式執行過程當中，在 p4 處理過程當中，想要把哪個欄位或什麼值印出來，只需要在前面加上 p4_logger，然後把想要印出來的東西放在後面，就可以察覺他們之間的變化

ip_forward.p4

```
#include <core.p4>
#include <v1model.p4>
typedef bit<48> macAddr_t;
typedef bit<9> egressSpec_t;


header arp_t {
        bit<16> htype;
        bit<16> ptype;
        bit<8>    hlen;
        bit<8>    plen;
        bit<16> opcode;
        bit<48> hwSrcAddr;
        bit<32> protoSrcAddr;
        bit<48> hwDstAddr;
        bit<32> protoDstAddr;
}


header ethernet_t {
        bit<48> dstAddr;
        bit<48> srcAddr;
        bit<16> etherType;
}


header ipv4_t {
        bit<4>    version;
        bit<4>    ihl;
        bit<8>    diffserv;
        bit<16> totalLen;
        bit<16> identification;
        bit<3>    flags;
        bit<13> fragOffset;
```

```
    bit<8>    ttl;

    bit<8>    protocol;
    bit<16> hdrChecksum;
    bit<32> srcAddr;
    bit<32> dstAddr;
}


struct metadata {

}


struct headers {
    @name(".arp")
    arp_t           arp;
    @name(".ethernet")
    ethernet_t ethernet;
    @name(".ipv4")
    ipv4_t          ipv4;
}


parser ParserImpl(packet_in packet, out headers hdr, inout metadata meta, inout
standard_metadata_t standard_metadata) {

    @name(".parse_arp") state parse_arp {
        packet.extract(hdr.arp);
        transition accept;
    }

    @name(".parse_ethernet") state parse_ethernet {

        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            16w0x800: parse_ipv4;
            16w0x806: parse_arp;
```

```
                default: accept;

            }
        }

        @name(".parse_ipv4") state parse_ipv4 {
            packet.extract(hdr.ipv4);
            transition accept;
        }

        @name(".start") state start {
            transition parse_ethernet;
        }
}


control egress(inout headers hdr, inout metadata meta, inout standard_metadata_t
standard_metadata) {

        apply {

        }
}


control ingress(inout headers hdr, inout metadata meta, inout standard_metadata_t
standard_metadata) {

        @name(".set_nhop") action set_nhop(macAddr_t dstAddr, egressSpec_t port) {
            //set the src mac address as the previous dst, this is not correct right?
            hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;


            //set the destination mac address that we got from the match in the table
            hdr.ethernet.dstAddr = dstAddr;


            //set the output port that we also get from the table
```

```
            standard_metadata.egress_spec = port;


            //decrease ttl by 1
            hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
        p4_logger(hdr.ipv4.ttl);
        p4_logger(hdr.ipv4.srcAddr);
        }

        @name("._drop") action _drop() {
            mark_to_drop(standard_metadata);
        }

        @name(".ipv4_lpm") table ipv4_lpm {
            actions = {
                set_nhop;
                _drop;
            }

            key = {
                hdr.ipv4.dstAddr: lpm;
            }

            size = 512;
            const default_action = _drop();
        }

        apply {
            ipv4_lpm.apply();
        }
}


control DeparserImpl(packet_out packet, in headers hdr) {

        apply {
            packet.emit(hdr.ethernet);
            packet.emit(hdr.arp);
```

在執行過程中，如果想知道某個欄位的值是多少，例如：執行過程中 ttl 會-1，這個 ttl 值是多少，就可以打上 p4_logger(值，這裡是 hdr.ipv4.ttl)，加上分號並儲存

若是想知道這個封包現在來源 ip 是多少，就可以把欄位放()中。
p4_logger(hdr.ipv4.srcAddr);

```
        packet.emit(hdr.ipv4);
    }
}


control verifyChecksum(inout headers hdr, inout metadata meta) {

    apply {
        verify_checksum(true, { hdr.ipv4.version, hdr.ipv4.ihl, hdr.ipv4.diffserv,
hdr.ipv4.totalLen, hdr.ipv4.identification, hdr.ipv4.flags, hdr.ipv4.fragOffset,
hdr.ipv4.ttl, hdr.ipv4.protocol, hdr.ipv4.srcAddr, hdr.ipv4.dstAddr },
hdr.ipv4.hdrChecksum, HashAlgorithm.csum16);
    }
}


control computeChecksum(inout headers hdr, inout metadata meta) {

    apply {
        update_checksum(true, { hdr.ipv4.version, hdr.ipv4.ihl, hdr.ipv4.diffserv,
hdr.ipv4.totalLen, hdr.ipv4.identification, hdr.ipv4.flags, hdr.ipv4.fragOffset,
hdr.ipv4.ttl, hdr.ipv4.protocol, hdr.ipv4.srcAddr, hdr.ipv4.dstAddr },
hdr.ipv4.hdrChecksum, HashAlgorithm.csum16);
    }
}


V1Switch(ParserImpl(), verifyChecksum(), ingress(), egress(), computeChecksum(),
DeparserImpl()) main;
```