```
    1: // Copyright 2017 Patrick Muldoon
    2: #ifndef INTOUCH_HPP_
    3: #define INTOUCH_HPP_
    4:
    5: #include <boost/regex.hpp>
    6: #include <iostream>
    7: #include <string>
    8: #include <vector>
    9:
   10: const std::string ServerStart =
   11:   "(\\d{4})\\-(\\d{2})\\-(\\d{2}) (\\d{2}):(\\d{2}):(\\d{2}): \\(log.c.16
6\\) server started\\s*";
   12:
   13: const std::string StartupSucceeded =
   14:   "(\\d{4})\\-(\\d{2})\\-(\\d{2}) (\\d{2}):(\\d{2}):(\\d{2})\\.(\\d{3}).*
oejs.AbstractConnector:Started"
   15:   " SelectChannelConnector.*";
   16:
   17: const std::string ServiceStarted =
   18:   "^Starting Service.\\s\\s(\\w+)\\s(1\\.[0-9]|1\\.[0-9]\\.[0-9]).*";
   19:
   20: const std::string SoftloadStart =
   21:  "^([A-Z?][a-z]{2})\\s(\\d{2})\\s(\\d{2}):(\\d{2}):(\\d{2}).*SOFTLOADSERV
ICE;Install started.*";
   22:
   23: const std::string SoftloadComplete =
   24:  "^([A-Z?][a-z]{2})\\s(\\d{2})\\s(\\d{2}):(\\d{2}):(\\d{2}).* ExitValue f
rom install command : 0.*";
   25:
   26: const std::string ServiceSucceeded =
   27:  "^Service started successfully.\\s\\s(\\w+)\\s(1\\.[0-9]|1\\.[0-9]\\.[0-
9])\\s\\((\\d+)\\s(\\w+)\\).*";
   28:
   29: const std::string OriginalVersion = "^([A-Z?][a-z]{2})\\s(\\d{2})\\s(\\d{
2}):(\\d{2}):(\\d{2}).*intouch-"
   30: "platform-base-(\\d)\\.(\\d)\\.(\\d)\\-(\\w+\\.\\d{2}|\\d{1,3})\\.armv6je
l_vfp.rpm.*rpm to rollback list";
   31:
   32: const std::string NewVersion = "^([A-Z?][a-z]{2})\\s(\\d{2})\\s(\\d{2}):(
\\d{2}):(\\d{2}).*intouch-platform"
   33: "-base-(\\d)\\.(\\d)\\.(\\d)\\-(\\w+\\.\\d{2}|\\d{1,3}).*armv6jel_vfp.rpm
\\s\\.\\.\\.";
   34:
   35: const boost::regex StartRegex(ServerStart);
   36: const boost::regex SucceededRegex(StartupSucceeded);
   37: const boost::regex ServiceStart(ServiceStarted);
   38: const boost::regex ServiceSuccess(ServiceSucceeded);
   39: const boost::regex SoftLoadBegin(SoftloadStart);
   40: const boost::regex SoftLoadEnd(SoftloadComplete);
   41: const boost::regex Original(OriginalVersion);
   42: const boost::regex New(NewVersion);
   43:
   44: class Services {
   45: public:
   46:         Services(std::string start_line, std::string _filename, unsigned
int line_number);
   47:         void ServiceBoot(std::string successful_line, unsigned int line_n
umber);
   48:         friend std::ostream& operator<<(std::ostream &out, const Services
 &service);
   49:         std::string getServiceName() {return service_name;};
   50:         bool getSuccess() {return success;};
   51: private:
   52:         std::string filename, service_name, boot_time;
   53:         bool success;
```

```
   54:         unsigned int start_line_number, end_line_number;
   55: };
   56:
   57: class Intouch{
   58:  public:
   59:         Intouch(std::string start_line, std::string _filename, unsigned i
nt line_number);
   60:         void BootSuccess(std::string successful_line, unsigned int line_n
umber);
   61:         friend std::ostream& operator<< (std::ostream &out, const Intouch
 &it);
   62:         std::vector<Services> a;
   63:
   64:         bool getSuccess() {return success;};
   65:         unsigned int getStartLine() {return start_line_number;};
   66:         unsigned int getEndLine() {return end_line_number;};
   67:  private:
   68:         unsigned int Time_Elapsed();
   69:         std::string filename, start_time, end_time;
   70:         bool success;
   71:         unsigned int start_line_number, end_line_number, boot_time;
   72: };
   73:
   74: class Softload {
   75: public:
   76:         Softload(std::string start_line, std::string _filename, unsigned
int line_number);
   77:         void Originalver(std::string successful_line);
   78:         void Newver(std::string successful_line);
   79:         void SoftloadSuccess(std::string successful_line, unsigned int li
ne_number);
   80:         unsigned int getStartLine() {return start_line_number;};
   81:         unsigned int getEndLine() {return end_line_number;};
   82:         std::string getStartTime() {return start_time_soft;};
   83:         std::string getEndTime() {return end_time_soft;};
   84:         std::string getFileName() {return filename;};
   85:         std::string getOriginal() {return oldSoftLoad;};
   86:         std::string getNew() {return newSoftLoad;};
   87:         std::string getBegin() {return begin;};
   88:         std::string getStop() {return stop;};
   89:         bool getSuccess() {return success;};
   90: private:
   91:         unsigned int Time();
   92:         std::string filename, start_time_soft, end_time_soft;
   93:         std::string begin, stop;
   94:         std::string oldSoftLoad, newSoftLoad;
   95:         bool success;
   96:         unsigned int start_line_number, end_line_number, boot_time;
   97: };
   98:
   99: #endif
```