DNNXPS

03월 07일 Lab Meeting

KHU spondlab
최 정 훈

DNNXPS

기존 문제점 1

(InitNum Error):
main.cpp(Generator)에서 생성한 랜덤한 식이 main.py(DNN)에서 읽는 과정에서 “InitNum”이 전달되는 오류 발생

```
1: configReader.cpp
2 #include "configReader.hpp"
1
3 const std::vector<std::pair<std::string, std::string>> defaultConfig{
1   {"InitNum", "0"},
2   {"TargetNum", "20000000"},
3   {"SessaReconnect", "1000"},
4   {"XPSsource", "ALKA"},
5   {"ChemRdnSft", "5"},
6   {"ContaRdnSft", "0.5"},
7 };
```

```
14 AtomRatio randomGenerator::Random(int N){
13   std::mt19937 gen(this->rd());
12 ..
11   AtomRatio atomratio(N);
10 ..
9    std::uniform_int_distribution<int> atom(0, this->atomsN-1);
8    std::uniform_int_distribution<int> rat(0, 100);
7    std::uniform_int_distribution<int> depth(0, 4000);  // maxDepth fix?
6 ..
5    for(int i=0; i<N; ++i){
4      int at=atom(gen);
3      if(find(atomratio.atom.begin(), atomratio.atom.end(), at)!=atomratio.atom.end()){
2        i--; continue;
1      }
27     atomratio.atom[i]=at;
1      atomratio.ratio[i]=rat(gen);
2    }
3 ..
4    double sum=0;
5    for(double r: atomratio.ratio) sum+=r;
6    for(int i=0; i<N; ++i) atomratio.ratio[i]/=sum;
7 ..
8    atomratio.contaDepth=static_cast<double>(depth(gen))/100;
9 ..
10   return atomratio;
11 }
```

>>> objdump –d main

```
0000000100010000 g     0f SECT   0a 0000 [.data] __ZN5ATOMS5atomsE
```

```
000000010000c528 g       1e SECT   09 0080 [__DATA_CONST.__const] __ZTI4Info
0000000100010280 l       0e SECT   0b 0000 [.bss] __ZL13defaultConfig
0000000100006914 g       0f SECT   01 0000 [.text]   Z9Terminatev
```

```
> 0x280
[1] 640
> 640/8
[1] 80
```

Random 개방향 고려 실수와 Stack Overflow에 의한 오류

-> 해결 完

DNNXPS

기존 문제점 2

Sessa로 전체 path가 입력되지 않고, 일정 레벨에서 fflush() 되어버리는 현상

해당 문제는 이전 프로세스에서도 발생하던 문제점임

```
30  this->mtx.lock();
29
28  fprintf(this->fp, "\\\n"); fflush(this->fp);
27  fprintf(fp, "\\PROJECT SAVE OUTPUT %s\n", this->pathmanager.getDumpPath().c_str()); fflush(fp);
26  fprintf(this->fp, "\\\n"); fflush(this->fp);
25
24  this->mtx.unlock();
```

프로세스(main.cpp)이 동작하는 동안 '멀티 프로세스'로 동작하는
문제 여부 발생 가능성 검토 및 디렉토리 경로 상에서 기존(초기 저장 값)과 다른 파일 청소를 수행하는 Cleaner 프로세스 추가 작성

-> 해결 完

DNNXPS

DNNXPS 시스템 설명

~/Desktop/CP/KHU_project/DNNXPS/.codes » ls -al
total 296
drwxr-xr-x 26 csian staff   832 Mar 6 20:42 .
drwxr-xr-x 15 csian staff   480 Mar 2 18:16 ..
-rw-r--r--@ 1 csian staff   8196 Mar 2 18:11 .DS_Store
-rw-r--r-- 1 csian staff   1386 Mar 2 18:03 CNCommunicator.cpp
-rw-r--r-- 1 csian staff   407 Mar 2 18:03 CNCommunicator.hpp
-rw-r--r-- 1 csian staff   852 Jan 26 15:44 IPC2cpp.py
-rw-r--r-- 1 csian staff   2787 Jan 26 14:30 IPC2py.cpp
-rw-r--r-- 1 csian staff   1906 Jan 26 16:59 IPC2py.hpp
drwxr-xr-x   5 csian staff   160 Feb 15 23:11 __pycache__
-rwxr-xr-x 1 csian staff 46964 Mar 2 17:45 cleaner
-rw-r--r-- 1 csian staff   2484 Mar 2 17:45 cleaner.cpp
-rw-r--r--@ 1 csian staff   1887 Mar 6 20:36 configReader.cpp
-rw-r--r-- 1 csian staff   1030 Jan 26 16:37 configReader.hpp
-rw-r--r-- 1 csian staff   2192 Jan 26 15:43 dnnModel.py
-rw-r--r-- 1 csian staff   1179 Jan 18 23:46 exception.cpp
-rw-r--r-- 1 csian staff   1655 Jan 26 17:44 exception.hpp
-rw-r--r-- 1 csian staff   2180 Jan 20 13:27 filesystem.cpp
-rw-r--r-- 1 csian staff   2561 Jan 26 17:18 filesystem.hpp
-rw-r--r--@ 1 csian staff   1505 Mar 6 20:31 main.cpp
-rwx------ 1 csian staff   2010 Feb 15 23:26 main.py
-rw-r--r-- 1 csian staff   1033 Mar 2 17:32 makefile
-rw-r--r--@ 1 csian staff   5615 Mar 6 20:40 sessa.cpp
-rw-r--r-- 1 csian staff   3330 Mar 2 18:02 sessa.hpp
-rw-r--r--@ 1 csian staff   3472 Mar 6 20:39 utility.cpp
-rw-r--r-- 1 csian staff   2892 Jan 26 17:43 utility.hpp
-rw-r--r-- 1 csian staff   3164 Feb 15 23:11 utility.py

csian@Bonita-MacBookPro

>>> wget https://csian98.github.io/assets/code/XPSDNN/codes.tar
>>> tar –xvf codes.tar

>>> cd .codes
>>> make
>>> make clean

DNNXPS

DNNXPS 시스템 설명

main

sessa.hpp

```
final : class
  [prototypes]
 -Reconnect(void)
 +Sessa(void)
 +~Sessa()
 +dataWrite(void)
 -eqTransform(const AtomRatio&)
 -insideCounter(void)
 +operator ()(void)
 -randomShift(void)
 -sessaConnect(void)
 -sessaDisconnect(void)
 -subRoutine(const SessaInfo&)
  [members]
 -cnCommunicator
 -communicator
 -configreader
 -fp
 -inCounter
 -mtx
 -pathmanager
 -prCounter
 -random

SessaInfo : struct
  [prototypes]
 +SessaInfo(std::string, double)
  [members]
 +contaDepth
 +equation
```

utility.hpp

```
Info : class
  [prototypes]
 +getN(void) const
 +reConnect(void)
 +readData(void)
 #skipLine(int)

PeakInfo : class
  [prototypes]
 +~PeakInfo(void)
 +PeakInfo(std::string)
 +isConta(int index) const
  [members]
 -PeakPath
 -beg
 -contaLabel
 -fin
 -lNum
 +override
 -override
 +override
 +override
 -peakN

RawInfo : class
  [prototypes]
 +~RawInfo(void)
 +RawInfo(std::string)
 +getRawAddr(void) const
 -z2oNorm(void)
  [members]
 -bLine
 -fin
 -override
 +override
 +override
 +override
 -raw
 -rawPath
 -sz

randomGenerator : class
  [prototypes]
 +AtomN(void)
 +Random(int)
 +randomGenerator(int)
 +~randomGenerator()
 +zeroMidRdn(double)
  [members]
 -atomsN
 -rd
```

exception.hpp

```
SessaException : class
  [prototypes]
 +SessaException(std::string_view)
 +~SessaException(void)
 #localTime(void) const
 #setMsg(std::string_view)
 +writeLog(std::string_view) const
  [members]
 #errMsg
 #logPath
 +override

errException : class
  [prototypes]
 +errException(std::string_view)
 +~errException(void)
  [members]
 +override

logException : class
  [prototypes]
 +logException(std::string_view)
 +~logException(void)
  [members]
 +override
```

filesystem.hpp

```
CheckFile : class
  [prototypes]
 +CheckFile(std::string)
 +~CheckFile()
 +CheckFile(const CheckFile&)
 +CheckFile()
 +check(void)
 +getFile(void) const
 +rmFile(void)
 +waitFile60(void)
  [members]
 -file
 -path

PathManager : class
  [prototypes]
 +~PathManager()
 +PathManager(void)
 +checkDumpFileExist(void)
 +checkSpcFileExist(void)
 +dumpClear(void)
 +getCurPath(void) const
 +getDumpFilePath(void) const
 +getDumpPath(void) const
 +getSessaLoad(void) const
 +getSpcFilePath(void) const
 +rmDumpFile(void)
 +rmSpcFile(void)
 +waitDumpFile60(void)
 +waitSpcFile60(void)
  [members]
 -checkDumpFile
 -checkSpcFile
 -curPath
 -dumpPath
 -sessaLoad
```

DNNXPS

DNNXPS 시스템 설명

main

IPC2py.hpp

```
Communicator : class
  [prototypes]
 +~Communicator(void)
 +Communicator(void)
 +exitPy(void)
 +initPy(void)
 +readAndWait(void)
 +writeInfo(std::string, double)
 +writeRaw(double* addr, int size=2048
  [members]
 -shrInfo
 -shrRaw

shrMemory : class
  [prototypes]
 -exitShr(void)
 -initShr(void)
 +readShr(void*)
 +~shrMemory(void)
 +shrMemory(const int key, const int m
 +writeShr(void*, int size=128)
  [members]
 -key
 -memSize
 -shmaddr
 -shmid

variables
  setKey
```

configReader.hpp

```
ConfigReader : class
  [prototypes]
 +ConfigReader(void)
 +~ConfigReader(void)
 +getValue(std::string) const
 -readConfig(void)
 -writeConfig(void)
  [members]
 -config
 -configFile
```

CNCommunicator.hpp

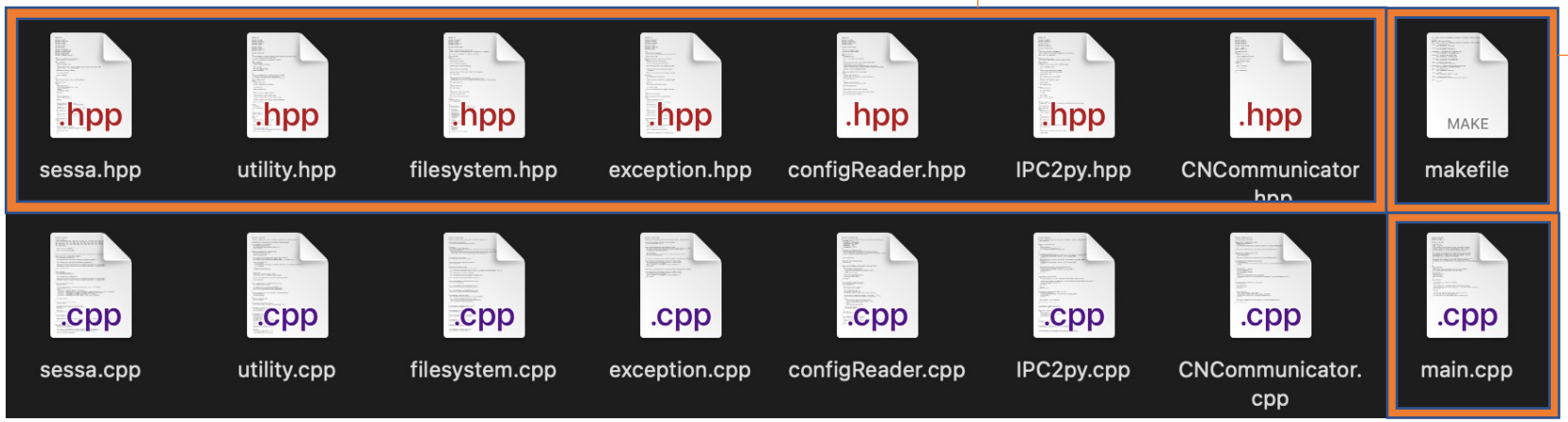
```
macros

CNCommunicator : class
  [prototypes]
 +~CNCommunicator(void)
 +CNCommunicator(void)
 +CNfork(std::string)
 +sendSignal(int&&)
  [members]
 -CNcount
 -path
 -pid
 -status
```

DNNXPS

DNNXPS 시스템 설명

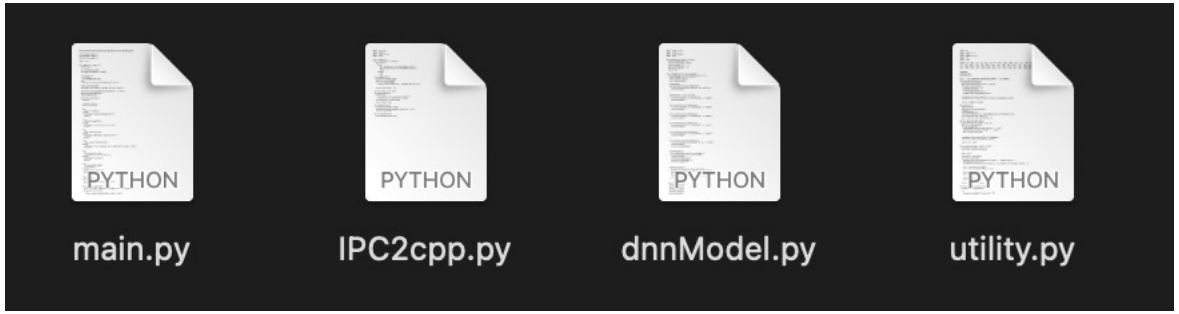
모든 주석(doxygen) header 파일에 작성!!!

main(ELF)

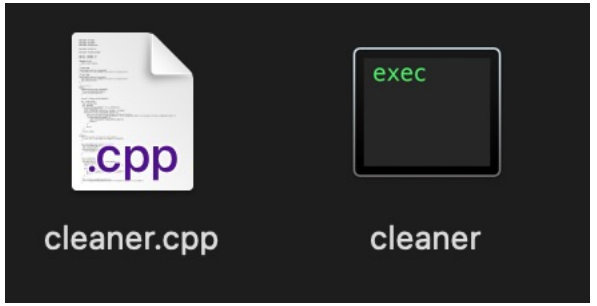


코드 의존관계 Makefile에 작성!!
./main
./.codes/cleaner

main.py (DNN)



cleaner(ELF)



DNNXPS

DNNXPS 시스템 설명

1: makefile

```
1  all : sessa.o utility.o configReader.o exception.o filesystem.o IPC2py.o CNCommunicator.o main.o ../main cleaner
1
2  CC = c++
3  CXXFLAGS = -Wall -O2 -std=c++2b
4  OBJS = sessa.o utility.o configReader.o exception.o filesystem.o IPC2py.o CNCommunicator.o main.o
5
6  sessa.o : sessa.hpp sessa.cpp
7  → $(CC) $(CXXFLAGS) -c sessa.cpp
8  →
9  utility.o : utility.hpp utility.cpp
10 → $(CC) $(CXXFLAGS) -c utility.cpp
11
12 configReader.o: configReader.hpp configReader.cpp
13 → $(CC) $(CXXFLAGS) -c configReader.cpp
14
15 exception.o : exception.hpp exception.cpp
16 → $(CC) $(CXXFLAGS) -c exception.cpp
17
18 filesystem.o : filesystem.hpp filesystem.cpp
19 → $(CC) $(CXXFLAGS) -c filesystem.cpp
20
21 IPC2py.o : IPC2py.hpp IPC2py.cpp
22 → $(CC) $(CXXFLAGS) -c IPC2py.cpp
23
24 CNCommunicator.o : CNCommunicator.hpp CNCommunicator.cpp
25 → $(CC) $(CXXFLAGS) -c CNCommunicator.cpp
26
27 main.o : sessa.hpp main.cpp
28 → $(CC) $(CXXFLAGS) -c main.cpp
29 →
30 ../main : $(OBJS)
31 → $(CC) $(CXXFLAGS) $(OBJS) -o ../main
32
33 cleaner : cleaner.cpp exception.o
34 → $(CC) $(CXXFLAGS) -o cleaner cleaner.cpp exception.o
35
36 clean :
37 → rm -f *.o
```

>>> make
>>> make clean

./main
./.codes/cleaner
두 개의 ELF file 생성

DNNXPS

DNNXPS 시스템 설명

main 사용법

include “sessa.hpp” <- 헤더 파일 인클루드(sessa.o 필요)

Sessa sessa ; Sessa 클래스 인스턴스 생성

sessa()
// or
sessa.dataWrite()

config::Target number 도달 시 false return
else true return

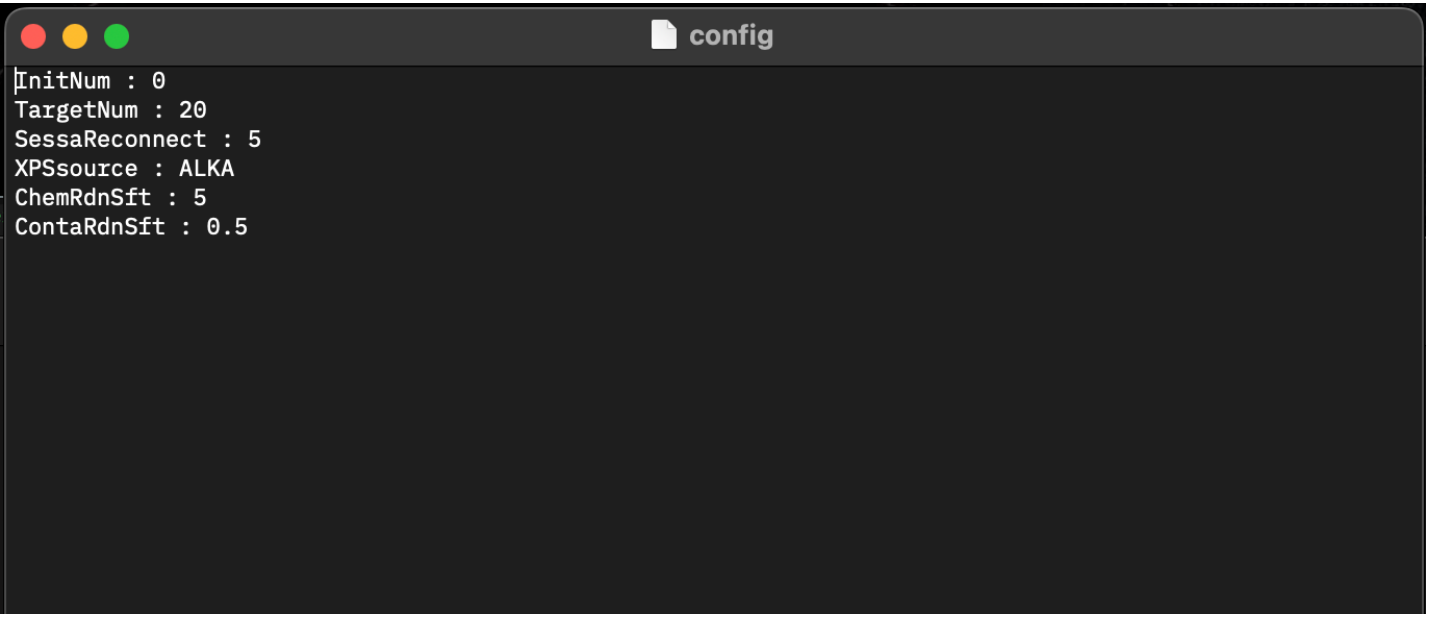
```cpp
#include "sessa.hpp"

int main(void){
  Sessa sessa;

  while(sessa()){}

  return 0;
}
```

```cpp
1: main.cpp
#include <iostream>
#include <string>
#include <string_view>

#include "sessa.hpp"

/* Copyright (C)
 * 2023 - sian.Choi
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 *
 */

/**
 * @file main.cpp
 * @brief //
 * Build random composition Atoms' XPS with sessa.exe(NIST)
 * and Training deep neural models with those data.
 *
 * KHU SpondLab
 * @author sian.Choi
 * @version 3.0.0
 * @date 2023-01-26
 */

int main(void){

  // Build Instance of Sessa to initialize
  Sessa sessa;

  // each sequence of 1) data Build and 2) deep neural training
  // progress with functor( Sessa::operator()  ) or method( Sessa::dataWrite() )
  while( sessa() ){}

  return 0;

  // process use diverse resources that require to be free
  // In case the process stops abnormally,
  // in particular, the shared memory must be freed directly.
}
```

DNNXPS

DNNXPS 시스템 설명

./config: 프로세스 시작 전에 읽고 설정



configReader.cpp & configReader.hpp

InitNum:
DNN 학습된 개수 지정(default 0)

TargetNum:
목표 학습 데이터 수(default 20,000,000)

SessaReconnect:
Sessa 재 연결 주기(default 1,000)
(Sessa는 재연결 하지 않을 시, 메모리 누적됨)

XPSsource:
광 종류(default AlKA)

ChemRdnSft:
Chemical Shift maximum(default 5)

ContaRdnShf:
Contamination Shift Maximum(default 0.5)

DNNXPS

DNNXPS 시스템 설명

./.tmp

```
~/Desktop/CP/KHU_project/DNNXPS/.tmp » tree
.
├── dump
├── log
│   ├── err.txt
│   ├── log.txt
│   └── pyLog.txt
└── losses
```

--- --- --- --- --- --- --- --- --- ---

./.tmp/dump:

SESSA exe 임시 디렉터리

./.tmp/log:

로그(cpp-log.txt, err.txt | py-pyLog.txt)
CTOR; constructor
CLN; Cleaner
DTOR; destructor
FORK; fork to main.py

./.tmp/losses:

utility.py :: validSpan 주기마다 validation Data에 대해 loss1.txt, loss2.txt 출력

err.txt

```
[2023/03/02 18:11:59] (CTOR) Initiate Sessa #0
[2023/03/02 18:11:59] (CLN) Initiate Cleaner: /Users/csian/Desktop/CP/KHU_project/DNNXPS/.tmp/dump/
[2023/03/02 18:13:22] (DTOR) Destruct Sessa #20
[2023/03/02 18:13:22] (CLN) Destruct Cleaner
```

log.txt

```
[2023/03/02 18:11:59] (FORK) Initiate main.py
[2023/03/02 18:11:59] (CNCommunicator) CNCommunicator Constructed
[2023/03/02 18:11:59] (CNCommunicator) CNCommunicator Build Complete
[2023/03/02 18:11:59] (CTOR) Initiate Seesa #0
[2023/03/02 18:11:59] (CLN) Initiate Cleaner: /Users/csian/Desktop/CP/KHU_project/DNNXPS/.tmp/dump/
[2023/03/02 18:12:23] (RECONN) Re-Integration of Sessa #5
[2023/03/02 18:12:44] (RECONN) Re-Integration of Sessa #10
[2023/03/02 18:13:00] (RECONN) Re-Integration of Sessa #15
[2023/03/02 18:13:18] (TARGET) Reach Target Data Count #20
[2023/03/02 18:13:22] (DTOR) Destruct Sessa #20
[2023/03/02 18:13:22] (CLN) CLN SIGUSR2 received
[2023/03/02 18:13:22] (CNCommunicator) CNCommunicator Destructed
[2023/03/02 18:13:22] (FORK) Exit main.py
[2023/03/02 18:13:22] (CLN) Destruct Cleaner
```

pyLog.txt

```
[Thu Mar  2 18:12:01 2023] ShrMemory Linked
[Thu Mar  2 18:12:01 2023] Initialize Programm from 0
[Thu Mar  2 18:13:22 2023] Terminate Process without Trouble Founded
```

DNNXPS

DNNXPS 시스템 설명

>>> ./main

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS
3227	main	185.9	02:16.90	2/2	0	16	2545K	0B	0B	3227	3013	running	*0[1]	0.00000	0.00000
3228	python3.8	104.6	01:30.25	23/1	7	66+	250M+	0B	0B	3227	3227	running	*0[1]	0.00000	0.00000
3244	SESSA v2.2	96.8	00:18.81	7/1	4	590+	324M+	86M-	0B	3227	3227	running	*0-[4]	0.85817	0.00349

+ cleaner(signal 받을 때까지 mutex lock; pause)
기타 파일 wait는 spin lock

```
~ » ipcs
IPC status from <running system> as of Mon Mar  6 22:19:19 KS
T     ID     KEY        MODE       OWNER    GROUP
Message Queues:

T     ID     KEY        MODE       OWNER    GROUP
Shared Memory:
m  65536 0x00000cca --rw-rw-rw-    csian    owner
m  65537 0x00000ccb --rw-rw-rw-    csian    owner

T     ID     KEY        MODE       OWNER    GROUP
Semaphores:
```

./main

- Sessa
  - Communicator
  - FILE* fp
  - ConfigReader
  - PathManager
  - randomGenerator
  - CNCommunicator

  prCounter
  inCounter
  std::mutex mtx

Shared memory
[multi thread;Sessa::subRoutine]

Pipe IPC
[Multi Process; SESSA.exec]

Signal processing:
SIGUSR1(wake up),
SIGUSR2(quit)
[Multi Process; cleaner]

./.codes/main.py

SESSA.EXEC

./.codes/cleaner

3274: raw data[double(8byte)X2048]
3275: info[state+Eqn+conta Depth]

cleaner: 오류 점검 및 회생 시도

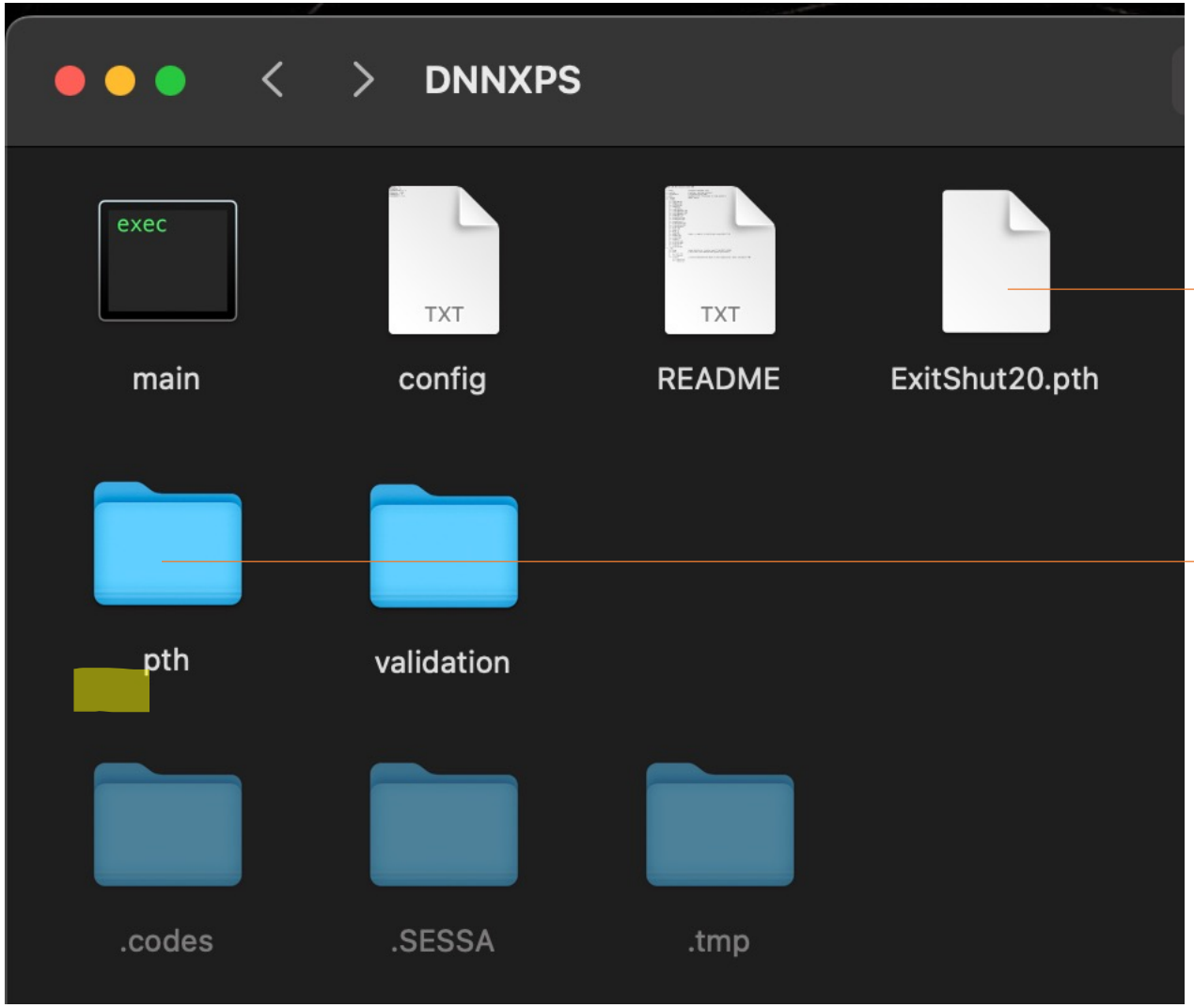
```
[2023/03/06 22:19:12] (CTOR) Initiate Sessa #0
[2023/03/06 22:19:13] (CLN) Initiate Cleaner: /Users/csian/Desktop/CP/KHU_project/DNNXPS/.tmp/dump/
[2023/03/06 22:21:52] (OVERTIME) Wait More than 60 sec
[2023/03/06 22:20:28] (RECONN) Re-Integration of Sessa #15
[2023/03/06 22:21:52] (CLN) CLN SIGUSR1 received
[2023/03/06 22:21:52] (CLN) CLN Operated
[2023/03/06 22:22:08] (RECONN) Re-Integration of Sessa #19
[2023/03/06 22:22:09] (TARGET) Reach Target Data Count #20
```

DNNXPS

DNNXPS 시스템 설명

>>> ./main

After reach target Number



종료 시 *.pth(DNN weights 파일) 생성

utility.py::validSpan마다 가중치 저장
(XPSdnn000000.pth, # 은 1000 per 1)