김 영 찬

근본을 중요시하고 납득될 때까지 물음표를 던지는 신입 개발자 김영찬입니다.

보다 나은 성능의 개선과 새로운 기술에는 근본적인 지식과 이해가 선행되어야 하고, 이는 수많은 기술들과 빠르게 변해가는 세상에서 무엇보다 중요한 것이라고 생각합니다.

동작 원리를 추측하고 기초지식과 이를 바탕으로 배우고 문제를 해결하려고 노력하는 개발자를 목표로 하고 있습니다.

○ 학력

2016 ~ 2023	인하대학교 공과대학 졸업
	ㄴ 사회인프라공학 전공

○ 어학

OPIC	날짜 : 2022.10.17 등급 : IH

○ 교육

2023 ~ 현재	42Seoul
	ㄴ Linux, Shell, C/C++ 프로젝트 기반 교수없는 자기주도학습을 통한 컴퓨터과학 학습

○ 기술

Language	●C ●C++ ●Python ●Assembly
Library / Framework	●Django
Tools	●Nginx ●Docker ●ELK(ElasticSearch/Logstash/Kibana)
Etc	●Linux ●LLDB

●원활한 활용
●사용가능
●경험있음

○ 프로젝트

프로젝트명	Webserv
과제목표	정적파일 서빙과 CGI처리가 가능한 웹서버 구현(Request파싱 및 CGI처리 담당)
내용	- TCP/IP Socket관련 시스템콜 함수를 이용해 HTTP프로토콜 웹서버 구현 - RFC문서를 통한 HTTP 프로토콜의 기본적인 이해 - Multiplexing I/O 모델을 사용하여 한번에 한 이벤트(read, write), 한번에 일정 버퍼사이즈만큼만 읽음으로써 서버의 공평한 처리가 가능 - 리퀘스트 파싱을 통한 HTTP 프로토콜 유효성 검사를 통해 각종 헤더들에 대해 알게 됨 - FAST-CGI 가 왜 나오게 되었는지에 대해 알게 됨 - TIMER 이벤트를 이용해 리퀘스트와 리스폰스, CGI가 오래걸릴때 타임아웃 처리 - 이벤트 fd를 통해 처리되므로 클라이언트 식별을 해야하는데, 찾는 시간을 줄이기위해 Map 컨테이너와 배열을 이용해 모든 자료 검색을 O(logn)에 가깝게 처리하여 테스터기 소요시간 줄임 - CGI와 리스폰스의 처리에서 string 객체를 통해 처리할 때 레퍼런스로 처리, 일정 버퍼 사이즈만큼 read/write를 하게 했으므로, 이미 완료한 부분을 잘라내는 식으로 한다면 복사가 일어나므로 따로 index 변수를 통해 처리
세부내용 링크 (Github)	https://github.com/dudcks0994/Webserv

프로젝트명	Transcendence
과제목표	Django와 VanilaJS를 이용한 핑퐁게임 웹사이트 구현 (Backend 및 Docker파트 담당)
내용	- 웹소켓 프로토콜에 대해 기본적인 이해, 비동기 프로그래밍 경험 - 웹소켓을 이용하여 친구의 온라인/오프라인 스테이터스를 구현 - OTP인증을 추가하기 위해, JWT + Session인증과 Session단일인증으로 구분하여 인가문제 해결 - Docker을 사용하여 Nginx를 프록시버로 백엔드와 프론트서버 연결 - self-signed TLS를 적용하여 HTTPS 통신 적용 - 세 개의 도커 컨테이너를 이용해 ELK스택을 구축하여 로그의 시각화 및 인덱스의 생명주기 설정
세부내용 링크 (Github)	https://github.com/dudcks0994/transcendence

프로젝트명	Philosopher
과제목표	멀티쓰레드/멀티프로세스로 기아회피 및 데드락방지하는 '식사하는 철학자' 문제 구현
내용	- 뮤텍스 / 세마포어를 통해 포크를 드는 행위의 원자성 보장 및 데이터레이스 방지 - 데드락 방지를 위해 집는 포크순서와 먼저먹는 순서 변경 - IPC가 사용이 불가능한 멀티프로세스 환경에서 먹는횟수를 공유하기 위해, 각각의 감시용 스레드를 생성한 뒤 먹는 루틴 사이클에 철학자 한명씩 세마포어를 만들어 post, wait하면서 카운트하게 처리 - 철학자가 죽은 이후 출력이 나오지 않게 출력 뮤텍스를 만들어 죽음체크-출력 의 원자성 보장
세부내용 링크 (Github)	https://github.com/dudcks0994/Philosoper

프로젝트명	Minishell
과제목표	기본적인 기능들을 구현한 미니 Shell 프로그램 만들기
내용	- POSIX 문서를 참고해 입력을 파싱하여 메타캐릭터를 기준으로 토큰화 - cmd 실행 및 heredoc의 fork()를 대비한 시그널 처리 - 프로세스 당 최대 fd제한이 있어, 부모프로세스의 모든것을 복사하는 fork()의 특성을 이용해 만 개 이상의 cmd도 잘 처리되게 구현 - 입력을 기다리며 종료되지 않는 프로그램의 경우 SIGPIPE를 이용해야 정상적으로 파이프가 목표대로 동작함 - 동적 메모리 할당과 해제를 통한 메모리 관리
세부내용 링크 (Github)	https://github.com/dudcks0994/minishell