시스템프로그래밍

과 제 명 : Proxy 3-2

교 수 님 : 최상호 교수님

강의시간 : 월요일

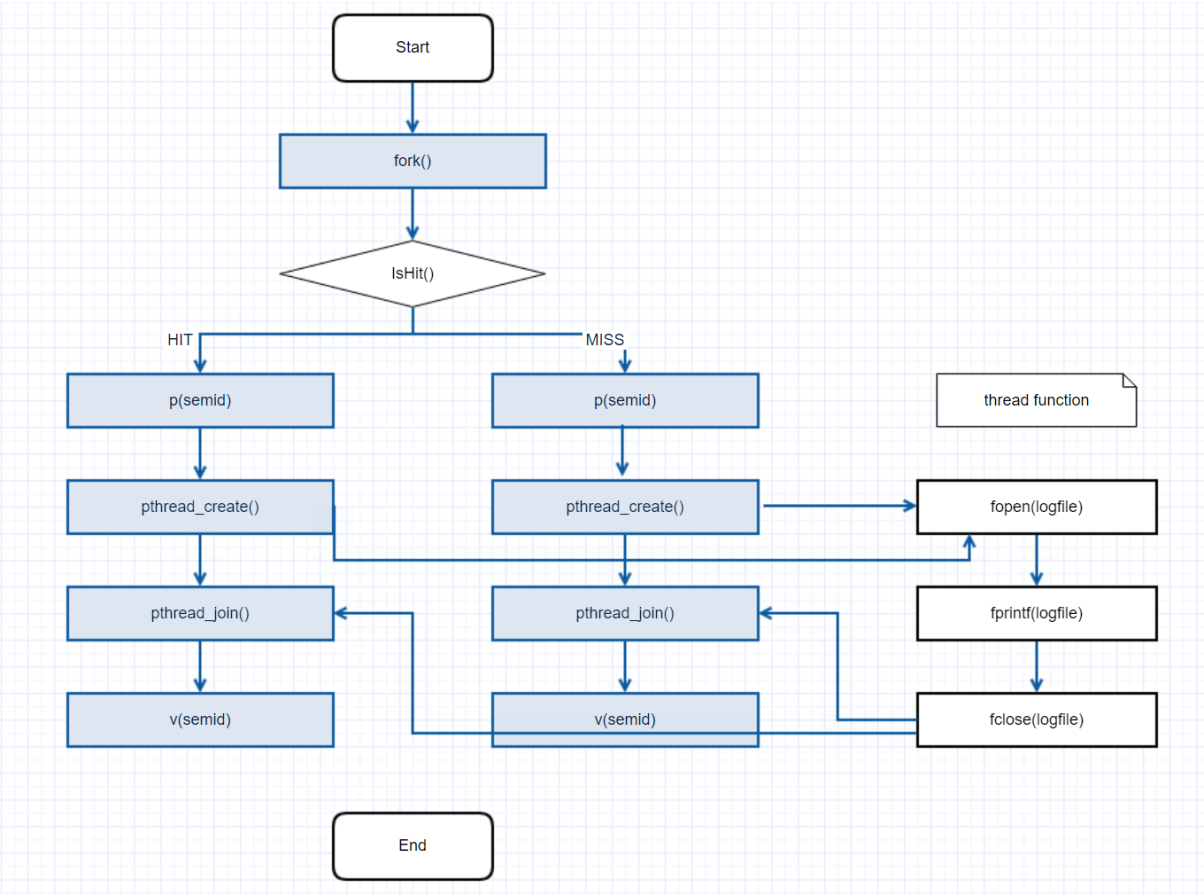
학 번 : 2019202100

이 름 : 하주영

Introduction

이번 과제에서 활용할 개념은 thread이다. Thread 함수는 fork와는 다르게 메모리 공간 중에서 stack과 register만 따로 생성하고 나머지 code, data, files는 공유하고 있기 때문에 fork함수에 비해 비용이 적게 드는 장점이 있다. 이전 과제에서는 fork함수를 사용하여 자식프로세스를 생성하면 자식 프로세스 안에서 logfile을 작성하였지만 이번 과제에서는 자식프로세스 내에 thread를 생성하여 thread function내에서 logfile을 작성하도록 수정한다.

Flow Chart


Start
fork()
IsHit()
HIT
MISS
p(semid)
p(semid)
thread function
pthread_create()
pthread_create()
fopen(logfile)
pthread_join()
pthread_join()
fprintf(logfile)
v(semid)
v(semid)
fclose(logfile)
End

Pseudo Code

/*main*/

```
Int main() {
	pid = fork();
	if(pid == 0) {
		int err;
		void *tret;
		pthread_t tid;
		if(IsHIT()) {
			p(semid);
			strcpy(buf, "HIT");
			pthread_create(&tid, NULL, thr_fn,(void*)buf);
			pthread_join(tid,&tret);
			v(semid);
		}
		else {
			p(semid);
			strcpy(buf, "MISS");
			err = pthread_create(&tid,NULL,thr_fn,(void*)buf);
			pthread_join(tid,&tret);
			v(semid);
		}
	}
}
```

/*thread function*/

```
void *thr_fn(void *buf) {
	printf("*PID# %d create the *TID# %ld\n", getpid(),pthread_self());
	logfile을 열기
	if(strcmp(buf,"HIT")==0) {
		logfile 작성
	}
	else if (strcmp(buf,"MISS") == 0) {
		logfile 작성
	}
	printf("*TID# %ld is exited.\n",pthread_self());
}
```

Result Screen

semaphore안에서 logfile을 작성하되, logfile이 thread function에서 작성되도록 수정.

```
kw2019202100@ubuntu:~/3-2$ ./proxy_cache
*PID# 3643 is waiting the semaphore.
*PID# 3643 is in the critical zone.
*PID# 3643 create the *TID# 139881083750144
*TID# 139881083750144 is exited.
PID# 3643 exited the critical zone.
*PID# 3646 is waiting the semaphore.
*PID# 3646 is in the critical zone.
*PID# 3646 create the *TID# 139881083750144
*TID# 139881083750144 is exited.
PID# 3646 exited the critical zone.
*PID# 3648 is waiting the semaphore.
*PID# 3648 is in the critical zone.
*PID# 3648 create the *TID# 139881083750144
*TID# 139881083750144 is exited.
PID# 3648 exited the critical zone.
*PID# 3653 is waiting the semaphore.
*PID# 3653 is in the critical zone.
*PID# 3653 create the *TID# 139881083750144
*TID# 139881083750144 is exited.
PID# 3653 exited the critical zone.
```

Logfile을 thread함수를 사용하여 thread function내에서 작성되도록 수정.

Logfile이 3-1과 같이 작성됨을 확인함.

```
kw2019202100@ubuntu:~/3-2$ cat ~/logfile/logfile.txt
[MISS]klas.kw.ac.kr -[2022/06/07, 22:56:02]
[MISS]klas.kw.ac.kr/favicon.ico -[2022/06/07, 22:56:02]
[HIT]3ef/9fd210fb8e00c8114ff978d282258ed8a48ea -[2022/06/07, 22:56:03]
[HIT]klas.kw.ac.kr
[HIT]040/586ec76bfc413b57b4d072158a57feb25cc07 -[2022/06/07, 22:56:04]
[HIT]klas.kw.ac.kr/favicon.ico
**SERVER**[Terminated] run time : 5 sec. #sub process: 4
```

Consideration

Pthread_create는 새로운 thread를 생성하는 함수이다. Pthread_create의 4개의 인자중 void* 에 대해 이해한 것에 대하여 설명한다.

void *(***start**)(void*)	: 특정 함수(**start**) 를 호출함으로써 thread가 시작
void ***arg**	: start 함수의 인자

Void는 리턴타입이 void이며 매개변수도 void, 즉 없다는 의미이다. Void* 는 포인터는 포인터인데 어떤 변수의 주소값을 담을 지 그 타입이 정해지지 않았다는 뜻이다. 따라서 현재의 타입은 없지만 캐스팅을 통해 그 타입을 바꿔줄 수는 있다. 즉, 없지만 그 어떤 포인터로도 변화가 가능한 만능 포인터가 된다는 뜻이다. 따라서 void 포인터는 함수의 인자로 종종 쓰인다.

```
else if (strcmp(buf,"MISS") == 0) {
        fprintf(log,"[%s]%s ", (char*)buf,url);
        fprintf(log, "-[%04d/%02d/%02d, %02d:%02d:%02d]₩n",ltp->tm_year+1900, ltp->tm_mon+1, ltp->tm_mday, ltp->tm_hour, ltp->tm_min, ltp->tm_sec);
}
```

위의 코드는 이번 과제의 thr_fn의 logfile 작성부분이며, 인자로 buf를 void 타입으로 넘겨주었지만 캐스팅을 통해 char* 로 변경하여 파일에 출력하는 것을 확인할 수 있다.

Reference

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=sharonichoya&logNo=220501081810