OS\_Project#3 Thread

2013012148 Lee Jaeil

**project3**

**1. thread\_create**

using code of fork() and exec(), we can make pthread\_create  
1) use alloproc to make kstack, kstack will store cpu's registers states.  
make new struct proc *np. and insert return value of allocproc into np  
2) use current proc's page directory. LWP don't make new page directory.  
this is the purpose of using LWP. this feature makes thread have lower cost. do "np->pgdir = proc->pgdir".  
3) using proc's parent to represent main thread.  
"np->parent = proc" means that proc is main thread of np.  
4) copy current process's cpu register's state to new proc's kstack which is created by allocuvm.  
5) set new proc's eip as start\_routine(function name) which is location which thread start from.  
6) set np->isThread as 1. isThread is struct proc's member variable which represent thread when its value is 1.  
7) using for loop when i start from0 to 63(less than NPROC). In for loop, when proc->t*[*i*](https://hconnect.hanyang.ac.kr/2017_ELE3021_13065/2017_ELE3021_2013012148/wikis/struct%20proc's%20member%20variable%20that%20represent%20how%20many%20thread%20is%20created%20in%20main%20thread)*is zero, insert one into proc->t[i], insert i into np->tid(struct proc's member variable which represents thread id), and insert i into \*thread.  
8) using allocuvm to allocate user stack. As there are several threads in main thread, location which is allocated for new thread will be on top of stacks.  
sz = allocuvm(np->pgdir, proc->sz + np->tid* PGSIZE, proc->sz + (np->tid + 1) \* PGSIZE;  
9) set np->ustack(struct proc's member variable which represents location of the thread's user stack) as sz+PGSIZE.  
10) change np->sz as sz(new user stack's top)  
11) push arg(this will be used for function's argument) and fake address into stack. this makes thread think main function is working now.  
sp = (char\*)sz; sp-=4; *(uint*)sp = (uint)Arg; sp-=4; *(uint*)sp = 0xffffffff;  
12) make new proc's esp point sp(point 0xfffffffff). thread will push and pop into location sp.  
13) copy current opened file name to new proc  
14) makess np's state RUNNABLE as scheduler pick the proc.

**2. thread\_exit**

this function is similar with exit function  
1) close all open files  
2) wake up parent(main thread). When main thread wakes up, it deallocate zombie thread's kstack and user stack.  
3) change proc's state as ZOMBIE and sched(). scheduler will select other proc.  
4) save the retarg(which is received by thread\_exit function's argument) into arr\_ret[p->tid](https://hconnect.hanyang.ac.kr/2017_ELE3021_13065/2017_ELE3021_2013012148/wikis/which%20is%20struct%20proc's%20member%20variable)

**3. thread\_join**

this function is similar with wait function  
1) search ptable by for-loop. find the proc whose parent is main thread(current proc) and which is thread.  
2) If the proc which pass previous condition is zomble state and the proc's tid is equal to thread argument, select the proc and deallocate its kstack and user stack.  
3) push proc->arr\_ret[p->tid] into \*retval(which is received by thread\_join function's argument)

**4. exit**

In exit function, we think two case. first case is that thread call the exit system call. second case is that process call the exit system call.  
1) First, if thread call the exit function, run for-loop to find proc which is thread and has same parent and not equal to caller proc.  
2) deallocate proc's(which is selected previous condition) kernel stack and user stack.  
3) change parent(main thread)'s state as zombie. close all open files in parent proc  
4) deallocate caller proc(thread)'s user stack. and wakeup main thread's parent process.  
5) Second, if process call the exit function, exit runs like original exit(but several things are different).  
6) close all open files.  
7) wakeup proc's parent  
8) search ptable by for-loop. check if proc's parent is same as caller proc. if proc's state is ZOMBIE, wakeup initproc.  
9) if proc is thread, deallocate proc's kernel stack and user stack.  
10) change caller proc's state as ZOMBIE and do sched.