```
int body(), goUmode();

PROC *kfork(char *filename)
{
  int i;
  int *ptable, pentry;
  char *addr;
  /*
  char *cp, *cq;

  char line[8];
  int usize1, usize;
  int *ustacktop, *usp;
  u32 BA, Btop, Busp;
  */
  PROC *p = dequeue(&freeList);
  if (p==0){
    kprintf("kfork failed\n");
    return (PROC *)0;
  }
  p->ppid = running->pid;
  p->parent = running;
  p->parent = running;
  p->status = READY;
  p->priority = 1;
  p->cpsr = (int *)0x10;

 // build p's pgtable
  p->pgdir = (int *)(0x600000 + (p->pid - 1)*0x4000);
  ptable = p->pgdir;
  // initialize pgtable
  for (i=0; i<4096; i++)
    ptable[i] = 0;
  pentry = 0x412;
  for (i=0; i<258; i++){
    ptable[i] = pentry;
    pentry += 0x100000;
  }

  ptable[2048] = 0x800000 + (p->pid - 1)*0x100000|0xC32;
  // ptable[2049] = 0x900000 + (p->pid - 1)*0x200000|0xC32;

  p->cpsr = (int *)0x10;    // previous mode was Umode

  // set kstack to resume to goUmode, then to VA=0 in Umode image
  for (i=1; i<29; i++)  // all 28 cells = 0
    p->kstack[SSIZE-i] = 0;
  p->kstack[SSIZE-15] = (int)goUmode;  // in dec reg=address
  p->ksp = &(p->kstack[SSIZE-28]);

  addr = (char *)(0x800000 + (p->pid - 1)*0x100000);

  load(filename, p);

  p->usp = (int *)VA(0x100000);

  p->kstack[SSIZE-1] = VA(0);

  enqueue(&readyQueue, p);

  kprintf("proc %d kforked a child %d: ", running->pid, p->pid);
  printQ(readyQueue);
  addChild(p, running->pid);
  return p;
}

int fork()
{
  int i;
  int *ptable, pentry;
  char *PA, *CA;
  PROC *p = dequeue(&freeList);
  // Check if dequeue worked, if not, return since no available procs left.
  if (p==0){
    kprintf("kfork failed\n");
    return (PROC *)0;
  }

  p->ppid = running->pid;
  printf("New Pid: %d\n", p->pid);
  p->parent = running;
  p->status = READY;
  p->priority = 1;
  // build p's pgtable
  p->pgdir = (int *)(0x600000 + (p->pid - 1)*0x4000);
  ptable = p->pgdir;
  // initialize pgtable
  for (i=0; i<4096; i++)
    ptable[i] = 0;
  pentry = 0x412;
  for (i=0; i<258; i++){
    ptable[i] = pentry;
    pentry += 0x100000;
  }

  ptable[2048] = 0x800000 + (p->pid - 1)*0x100000|0xC32;
  PA = (char*)(running->pgdir[2048] & 0xFFFF0000);
  printf("PA pgdir[2048] = %x\n", PA);
  CA = (char*)(p->pgdir[2048] & 0xFFFF0000);
  printf("CA pgdir[2048] = %x\n", CA);
  memcpy((char*)CA, (char*)PA, 0x100000);
  for(i = 1; i <= 14; i++)
  {
    p->kstack[SSIZE - i] = running->kstack[SSIZE - i];
  }

  p->kstack[SSIZE - 14] = 0;
  p->kstack[SSIZE - 15] = (int)goUmode;
  p->ksp = &(p->kstack[SSIZE-28]);
  p->usp = running->usp;
  p->cpsr = running->cpsr;
  enqueue(&readyQueue, p);

  kprintf("proc %d kforked a child %d: ", running->pid, p->pid);
  printQ(readyQueue);
  addChild(p, running->pid);
  return p->pid;
}

int exec(char *cmdline)
{
  kprintf("line=%s\n", cmdline);
  int i, upa, usp;
  char *cp, kline[128], file[32], filename[32];
  PROC *p = running;
  strcpy(kline, cmdline);
  cp = kline;
  i = 0;
  while(*cp != ' ')
  {
    filename[i] = *cp;
    i++;
    cp++;
  }
  filename[i] = '\0';
  file[0] = '\0';
  if(filename[0] != '/')
  {
    strcpy(file, "/bin/");
  }
  kstrcat(file, filename);
  upa = p->pgdir[2048] & 0xFFFF0000;
  if(!load(file, p))
  {
    return -1;
  }
  usp = upa + 0x100000 - 128;
  strcpy((char*)usp, kline);
  p->usp = (int *)VA(0x100000 - 128);
  for(i = 2; i < 14; i++)
  {
    p->kstack[SSIZE - i] = 0;
  }
  p->kstack[SSIZE - 1] = (int)VA(0);
  printf("fork print %s\n", (char*)p->usp);
  return (int)p->usp;
}
```