COMP3000 - Exercise 2 (Individual)
Process - Process Table - Process Control Block - Inter Process Communication

Winter 2018

In this exercise, you implement a new system call with signature:

```
int <your 1st name here>_getdpids(pid_t top, pid_t dpids[], int len);
```

It finds process IDs (PIDs) of descendant processes of a given process, in breath first search order. The system call takes three arguments: the PID of the top process (`top`), a array to store the PIDs (`dpids[]`) and an integer that indicates the size of the array (`len`), in PIDs.

Test your system call with the following programs (must be updated with your 1st name).
Test program 1:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
int main(int argc, char** args) {
   pid_t pid, cpid; /* PIDs */
   pid_t dpids[LEN]; /* buffer for descendant PIDs */
   int i, r; /* control vars */
   printf("Test 1 - Single child\n");
   /* get the PID */
   pid = getpid();
   /* print the PID */
   printf("My PID is %d\n", pid);
   cpid = fork();
   if (cpid!=0) { /* in parent! */
      printf("Child PID is %d\n", cpid);
      /* get and print all descendants */
      r = michel_getdpids(pid, dpids, LEN);
```

```
      printf("Tree has %d process(es)\n", r);
      for (i=0;i<r;i++) { printf("%d ", dpids[i]); };
      printf("\n");
   } else { /* in child! */
      sleep(10); /* sleep 10 seconds */
   }
   return 0;
}
```

Test program 2:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#include <stdlib.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
#define N 5 /* number of forked children */
int main(int argc, char** args) {
   pid_t pid, cpid; /* PIDs */
   pid_t dpids[LEN]; /* buffer for descendant PIDs */
   int i, j, r; /* control vars */
   printf("Test 2 - Fan of children\n");
   /* get the PID */
   pid = getpid();
   /* print the PID */
   printf("My PID is %d\n", pid);
   /* fork N children */
   for (j=0; j<N;j++) {
      if (fork()<=0) /* in child! */
         break;
   }
   if (pid==getpid()) { /* in root! */
      /* get and print all descendants */
      r = michel_getdpids(pid, dpids, LEN);
      printf("Tree has %d process(es)\n", r);
      for (i=0;i<r;i++) { printf("%d ", (int) dpids[i]); };
      printf("\n");
      /* corroborate using "ps" */
      system("ps -o pid,ppid,command");
      } else { /* in child! */
         sleep(10);  /* sleep 10 seconds */
      }
return 0;
}
```

Test program 3:

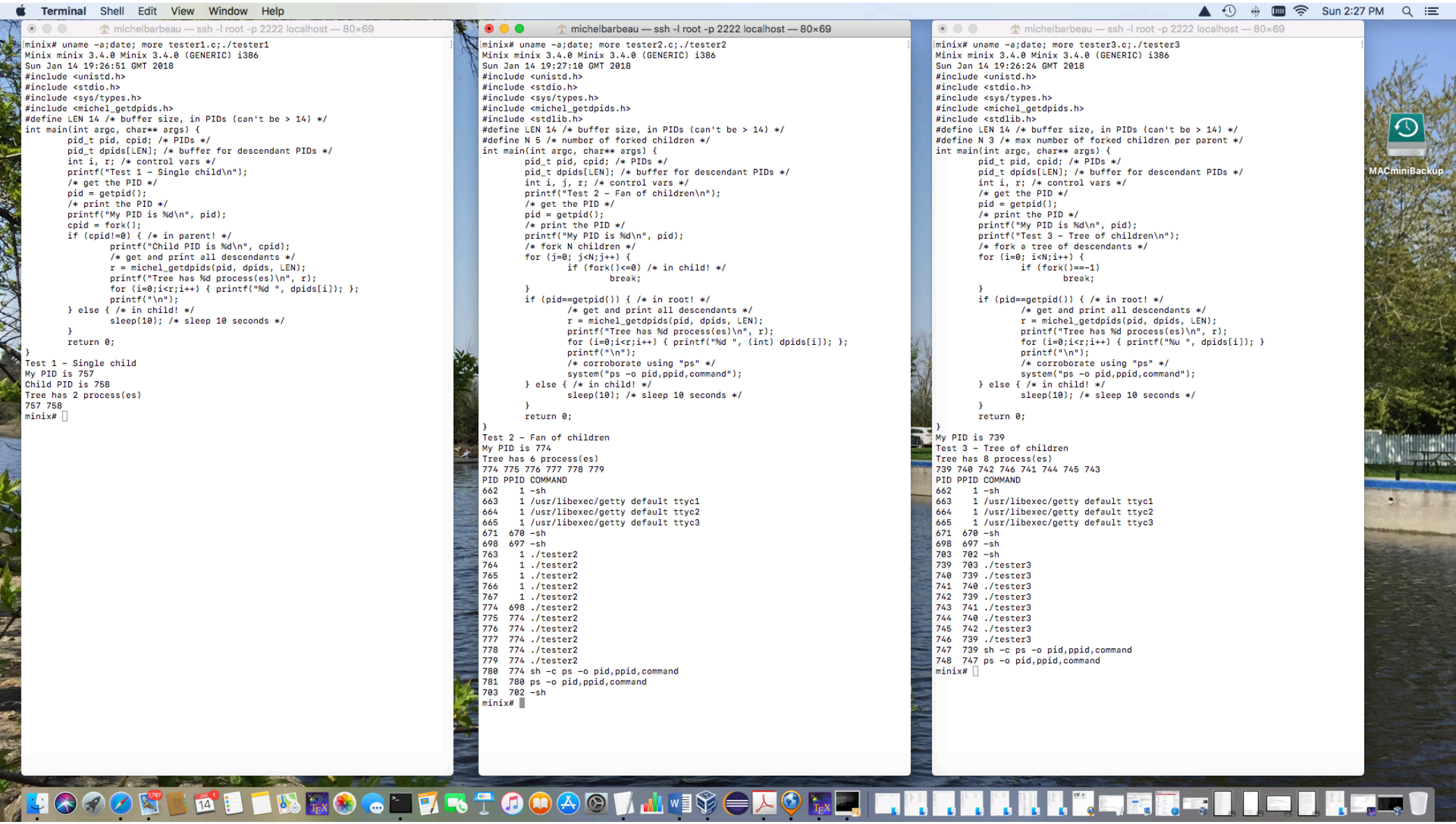
```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#include <stdlib.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
#define N 3 /* max number of forked children per parent */
int main(int argc, char** args) {
   pid_t pid, cpid; /* PIDs */
   pid_t dpids[LEN]; /* buffer for descendant PIDs */
   int i, r; /* control vars */
   /* get the PID */
   pid = getpid();
   /* print the PID */
   printf("My PID is %d\n", pid);
   printf("Test 3 - Tree of children\n");
   /* fork a tree of descendants */
   for (i=0; i<N;i++) {
      if (fork()==-1)
         break;
   }
   if (pid==getpid()) { /* in root! */
      /* get and print all descendants */
      r = michel_getdpids(pid, dpids, LEN);
      printf("Tree has %d process(es)\n", r);
      for (i=0;i<r;i++) { printf("%u ", dpids[i]); }
      printf("\n");
      /* corroborate using "ps" */
      system("ps -o pid,ppid,command");
   } else { /* in child! */
      sleep(10); /* sleep 10 seconds */
   }
   return 0;
}
```

Assuming source code are in files `tester1.c`, `tester2.c` and `tester3.c` sample output:

```
minix# ./tester1
Test 1 - Single child
My PID is 692
Child PID is 693
Tree has 2 process(es)
```

```
692 693
minix# ./tester2
Test 2 - Fan of children
My PID is 694
Tree has 6 process(es)
694 695 696 697 698 699
PID PPID COMMAND
659    1 /usr/libexec/getty default console
660    1 /usr/libexec/getty default ttyc1
661    1 /usr/libexec/getty default ttyc2
662    1 /usr/libexec/getty default ttyc3
664  663 -sh
693    1 ./tester1
694  664 ./tester2
695  694 ./tester2
696  694 ./tester2
697  694 ./tester2
698  694 ./tester2
699  694 ./tester2
700  694 sh -c ps -o pid,ppid,command
701  700 ps -o pid,ppid,command
minix# ./tester3
My PID is 702
Test 3 - Tree of children
Tree has 8 process(es)
702 703 705 709 704 707 708 706
PID PPID COMMAND
659    1 /usr/libexec/getty default console
660    1 /usr/libexec/getty default ttyc1
661    1 /usr/libexec/getty default ttyc2
662    1 /usr/libexec/getty default ttyc3
664  663 -sh
702  664 ./tester3
703  702 ./tester3
704  703 ./tester3
705  702 ./tester3
706  704 ./tester3
707  703 ./tester3
708  705 ./tester3
709  702 ./tester3
710  702 sh -c ps -o pid,ppid,command
711  710 ps -o pid,ppid,command
```

Due date: January 28. This exercise must be done in the C programming language under MINIX 3.4. Submit your work on cuLearn. Submit a single *tar.gz* file. Include a README.txt file containing a report about your work (describe every change you made to the system code and where you made it), system call implementation and test program. Your submission must include a screenshot showing evidence that you code is working (see the attached example). Your are responsible for the completeness of your submission. Source code and a make file must be included. Submissions that do not compile are not accepted.


Terminal Shell Edit View Window Help
Sun 2:27 PM
michelbarbeau — ssh -l root -p 2222 localhost — 80×69
[minix# uname -a;date; more tester1.c;./tester1
Minix minix 3.4.0 Minix 3.4.0 (GENERIC) i386
Sun Jan 14 19:26:51 GMT 2018
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
int main(int argc, char** args) {
pid_t pid, cpid; /* PIDs */
pid_t dpids[LEN]; /* buffer for descendant PIDs */
int i, r; /* control vars */
printf("Test 1 - Single child\n");
/* get the PID */
pid = getpid();
/* print the PID */
printf("My PID is %d\n", pid);
cpid = fork();
if (cpid!=0) { /* in parent! */
printf("Child PID is %d\n", cpid);
/* get and print all descendants */
r = michel_getdpids(pid, dpids, LEN);
printf("Tree has %d process(es)\n", r);
for (i=0;i<r;i++) { printf("%d ", dpids[i]); };
printf("\n");
} else { /* in child! */
sleep(10); /* sleep 10 seconds */
}
return 0;
}
Test 1 - Single child
My PID is 757
Child PID is 758
Tree has 2 process(es)
757 758
minix#
michelbarbeau — ssh -l root -p 2222 localhost — 80×69
[minix# uname -a;date; more tester2.c;./tester2
Minix minix 3.4.0 Minix 3.4.0 (GENERIC) i386
Sun Jan 14 19:27:10 GMT 2018
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#include <stdlib.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
#define N 5 /* number of forked children */
int main(int argc, char** args) {
pid_t pid, cpid; /* PIDs */
pid_t dpids[LEN]; /* buffer for descendant PIDs */
int i, j, r; /* control vars */
printf("Test 2 - Fan of children\n");
/* get the PID */
pid = getpid();
/* print the PID */
printf("My PID is %d\n", pid);
/* fork N children */
for (j=0; j<N;j++) {
if (fork()<=0) /* in child! */
break;
}
if (pid==getpid()) { /* in root! */
/* get and print all descendants */
r = michel_getdpids(pid, dpids, LEN);
printf("Tree has %d process(es)\n", r);
for (i=0;i<r;i++) { printf("%d ", (int) dpids[i]); };
printf("\n");
/* corroborate using "ps" */
system("ps -o pid,ppid,command");
} else { /* in child! */
sleep(10); /* sleep 10 seconds */
}
return 0;
}
Test 2 - Fan of children
My PID is 774
Tree has 6 process(es)
774 775 776 777 778 779
PID PPID COMMAND
662 1 -sh
663 1 /usr/libexec/getty default ttyc1
664 1 /usr/libexec/getty default ttyc2
665 1 /usr/libexec/getty default ttyc3
671 670 -sh
698 697 -sh
763 1 ./tester2
764 1 ./tester2
765 1 ./tester2
766 1 ./tester2
767 1 ./tester2
774 698 ./tester2
775 774 ./tester2
776 774 ./tester2
777 774 ./tester2
778 774 ./tester2
779 774 ./tester2
780 774 sh -c ps -o pid,ppid,command
781 780 ps -o pid,ppid,command
703 702 -sh
minix#
michelbarbeau — ssh -l root -p 2222 localhost — 80×69
[minix# uname -a;date; more tester3.c;./tester3
Minix minix 3.4.0 Minix 3.4.0 (GENERIC) i386
Sun Jan 14 19:26:24 GMT 2018
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <michel_getdpids.h>
#include <stdlib.h>
#define LEN 14 /* buffer size, in PIDs (can't be > 14) */
#define N 3 /* max number of forked children per parent */
int main(int argc, char** args) {
pid_t pid, cpid; /* PIDs */
pid_t dpids[LEN]; /* buffer for descendant PIDs */
int i, r; /* control vars */
/* get the PID */
pid = getpid();
/* print the PID */
printf("My PID is %d\n", pid);
printf("Test 3 - Tree of children\n");
/* fork a tree of descendants */
for (i=0; i<N;i++) {
if (fork()==-1)
break;
}
if (pid==getpid()) { /* in root! */
/* get and print all descendants */
r = michel_getdpids(pid, dpids, LEN);
printf("Tree has %d process(es)\n", r);
for (i=0;i<r;i++) { printf("%u ", dpids[i]); }
printf("\n");
/* corroborate using "ps" */
system("ps -o pid,ppid,command");
} else { /* in child! */
sleep(10); /* sleep 10 seconds */
}
return 0;
}
My PID is 739
Test 3 - Tree of children
Tree has 8 process(es)
739 740 742 746 741 744 745 743
PID PPID COMMAND
662 1 -sh
663 1 /usr/libexec/getty default ttyc1
664 1 /usr/libexec/getty default ttyc2
665 1 /usr/libexec/getty default ttyc3
671 670 -sh
698 697 -sh
703 702 -sh
739 703 ./tester3
740 739 ./tester3
741 740 ./tester3
742 739 ./tester3
743 741 ./tester3
744 740 ./tester3
745 742 ./tester3
746 739 ./tester3
747 739 sh -c ps -o pid,ppid,command
748 747 ps -o pid,ppid,command
minix#
MACminiBackup