Julia Rieger and Rachel Nguyen

SET/GET PROCESS GROUP functions

#include <sys/types.h>
#include <unistd.h>

NOTE:

- **preferred way to get + set process group ID of a process is through getpgrp(void) and setpgid()**
- **Child created via fork inherits parent's PGID**
- **Only 1 process group can be the foreground process group for the terminal**
- **Signals generated from the terminal are sent to the whole foreground process group**

getpgrp()

pid_t getpgrp(void); //POSIX.1 ver. (more preferred)

Takes no args, returns PGID of the calling process
Retrieves the calling process's PGID

pid_t getpgrp(pid_t pid); //BSD version

Since glibc 2.19 the BSD version is no longer exposed by <unistd.h>, so we should use the POSIX.1 version instead

```
main() {
 int status;

 if (fork() == 0) { //in child
   if (fork() == 0) { //in grandchild
     printf("grandchild's pid is %d, process group id is %d\n", (int) getpid(), (int)
getpgrp());
     exit(0);
   }
   //in child
   printf("child's pid is %d, process group id is %d\n", (int) getpid(), (int)
getpgrp());
   wait(&status);
   exit(0);
```

```
 }
 //in parent/grandparent
 printf("parent's pid is %d, process group id is %d\n", (int) getpid(), (int)
getpgrp());
 printf("the parent's parent's pid is %d\n", (int) getppid());
 wait(&status);
}
```

OUTPUT (on my laptop)

```
parent's pid is 642798, process group id is 642798
the parent's parent's pid is 642529
child's pid is 642799, process group id is 642798
grandchild's pid is 642800, process group id is 642798
```

setpgrp()

int setpgrp(void); //System V ver.
int setpgrp(pid_t pid, pid_t pgid); //BSD version

Since glibc 2.19 setpgrp() is no longer exposed by <unistd.h>, so we should use setpgid() instead

Returns 0 on success and -1 on error (errno set accordingly)

getpgid()

```
pid_t getpgid(pid_t pid);
```

Usage: Get the process group ID of the process specified by pid. If pid is 0, getpgid() returns the PID of the calling process.
Return value:

- A process group on success
- -1 on error

Error:

- ESRCH: pid does not match any process

Sample code:

```
int main() {
```

```
    pid_t pgid = getpgid(0); // Get the PGID of the calling
process
    printf("Process Group ID: %d\n", pgid);
    return 0;
}
```

setpgid()

```
int setpgid(pid_t pid, pid_t pgid);
```

Usage: set the process group ID of the process specified by pid to pgid, so you can reassign a process to a different process group
Return value:

- 0 on success
- -1 on error

Errors:

- EACCES

The value of pid matches the PID of a child of the calling process, but the child has successfully run one of the EXEC functions.

- EINVAL

pgid is less than zero or has some other unsupported value.

- EPERM

The caller cannot change the PGID of the specified process

- ESRCH

pid does not match the PID of the calling process or any of its children.
Sample code:

```
int main() {
   int status;
   pid_t pid = fork();
   if (pid < 0) {
       perror("fork");
       return 1;
   } else if (pid == 0) {
       if (fork() == 0) {
           printf("In grandchild: Process ID: %d, Process Group ID: %d\n", getpid(),
getpgid(0));
           exit(0);
       }
       printf("In child: Process ID: %d, Process Group ID: %d\n", getpid(),
getpgid(0));
       printf("After setting child's PGID to its own PID\n");
```

```
        setpgid(0, 0);
        printf("Now in child: Process ID: %d, Process Group ID: %d\n", getpid(),
getpgid(0));

        exit(0);
    } else {
        printf("In parent: Process ID: %d, Process Group ID: %d\n", getpid(),
getpgid(0));
        wait(&status);
        exit(0);
    }
}
```

Output:

```
In parent: Process ID: 280376, Process Group ID: 280376
In child: Process ID: 280377, Process Group ID: 280376
After setting child's PGID to its own PID
Now in child: Process ID: 280377, Process Group ID: 280377
In grandchild: Process ID: 280378, Process Group ID: 280376
```