Assignment 3 Report

Q1b

Investigation

- The number of child processes created is `r1 * c2`.
- For N = 200, `fork()` returned an error and the program exitted. `errno` was set to `EAGAIN`.
- From the manpage of `fork()`:

  ```
  EAGAIN A system-imposed limit on the number of threads was
                encountered.  There are a number of limits that may
                trigger this error:

                *  the RLIMIT_NPROC soft resource limit (set via
                   setrlimit(2)), which limits the number of processes and
                   threads for a real user ID, was reached;

                *  the kernel's system-wide limit on the number of
                   processes and threads, /proc/sys/kernel/threads-max,
                   was reached (see proc(5));

                *  the maximum number of PIDs, /proc/sys/kernel/pid_max,
                   was reached (see proc(5)); or

                *  the PID limit (pids.max) imposed by the cgroup "process
                   number" (PIDs) controller was reached.
  ```

  - `ulimit -u` gave **`30798`** , which is the limit for number of processes and threads for a real user ID.
  - `cat /proc/sys/kernel/pid_max` gave **`4194304`**.
  - `cat /proc/sys/kernel/threads-max` gave **`61597`**, which is the system-wide limit for number of processes and threads. Not an issue either.
  - A `cgroup` is a collection of processes that are bound to a set of limits or parameters defined via the `cgroup` filesystem. As we are working on a Ubuntu 20.04 system, our Linux is running on `systemd`. So we checked the limits imposed on the user-slice by the process number controller.

    `cat /sys/fs/cgroup/pids/user.slice/user-$(id -u).slice/pids.max` gave **`20327`**!! **Clearly, this is the limiting value (least out of the four).**

Calculation of maximum size of matrix that can be multiplied (N)

- The number of processes currently in the cgroup is given by `pids.current`.
- `cat /sys/fs/cgroup/pids/user.slice/user-$(id -u).slice/pids.current` returns 940 (on an average) before the start of our program.

- `pids.max` - `pids.current` = 20327 - 940 = 19387 = Number of processes that our program can fork
- **Thus, max dimension of matrices =** N **=** $\sqrt{19387} \approx 139$.

Experiment

- We executed the the program several times with varying dimensions r_1 and c_2 (keeping $r_1 = c_2 = N$) for matrices A and B. We manually binary searched between 0 and 174 for the largest N for which the program ran successfully. Largest N was found to be **139**. So 139*139 = **19321** processes are being forked successfully, but not 140*140 = 19600. This indeed matches with the theoretical maximum.

References

- https://man7.org/linux/man-pages/man2/fork.2.html
- https://man7.org/linux/man-pages/man3/ulimit.3.html
- https://man7.org/linux/man-pages/man7/cgroups.7.html
- https://www.kernel.org/doc/Documentation/cgroup-v1/pids.txt#:~:text=The%20process%20number%20controller%20is,PIDs%20are%20a%20fundamental%20resource
- https://stackoverflow.com/questions/62180990/how-to-increase-number-of-child-proceses